IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC.,	)	
Plaintiffs/Counterclaim Defendants,	)	C.A. No. 06-414-SLR
v.	)	**JURY TRIAL DEMANDED**
EPICREALM LICENSING, LP,	)	**PUBLIC VERSION**
Defendant/Counterclaim Plaintiff.	)	

APPENDIX IN SUPPORT OF EPICREALM'S MOTION FOR PARTIAL SUMMARY JUDGMENT THAT THE ORACLE REFERENCES DO NOT ANTICIPATE

VOLUME I of IV

EXHIBITS 1 to 4

Richard L. Horwitz (#2246)
David E. Moore (#3983)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19899
Tel: (302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

Attorneys for Defendant/Counterclaim Plaintiff epicRealm Licensing, LP

OF COUNSEL:

Harry J. Roper
George S. Bosy
Aaron A. Barlow
Patrick L. Patras
David R. Bennett
Paul D. Margolis
Benjamin J. Bradford
Emily C. Johnson
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603
Tel: (312) 923-8305

Dated: July 31, 2008
Public Version Dated: August 7, 2008
876824 / 31393 / Oracle

TABLE OF CONTENTS

Exh.	Document
1.	U.S. Patent 5,894,554, Lowery et al.
2.	U.S. Patent 6,415,335, Lowery et al.
3.	Report of Plaintiffs' Expert Michael I. Shamos Concerning Invalidity, May 2, 2008
4.	Exhibit 3 from Shamos Expert Report
5.	Oracle WebServer User's Guide, Release 1.0 A34986-2, ORCL00941713-00941960, Ref. 153
6.	The Oracle WebServer: A Technical Discussion, ORCL01042968-01042974, Ref. 503
7.	Oracle WebServer Release 2.0, Feature List, 10/30/95, ORCL00937853-00937862
8.	Oracle WebServer 2.0 Technical Note March 1996, ORCL000761-000781, Ref 605
9.	Oracle WebServer User's Guide, Release 2.0 A23646-2, ORCL000782-1035, Ref. 604
10.	Parties' Proposed Element-by-Element Claim Construction Chart, June 30, 2008
11.	Excerpt from Oracle Internet Directory Application Developer's Guide, Release 2.1.1, September 2000, A86082-01, ORCL00106573-00106712
12.	U.S. Patent 5,761,673 Bookman, et al.
13.	The Rush to the Net, May 1996, ORCL01606514-01606526
14.	Oracle World, ORCL01050220-01050280
15.	InfoSpinner Status, 11/15/95, EPIC000146-000148
16.	Intentionally Left Blank
17.	WebSpinner product summary, EPIC000168-000173
18.	Distribution Agreement between Software AG of North America and InfoSpinner, 2/28/96, EPIC000006-000032
19.	Excerpt from Understanding SQL*Net Release 2.2, ORCL01806620-01806818
20.	Excerpt from Functional Specification for Runtime Load Balancing, RDBMS, 10gR2, ORCL01060148-01060193

Exh.	Document
21.	Excerpt from Understanding SQL Net Release 2.3, A42484-1, ORCL01808597-01808820
22.	USSN 09/234,048, US 6,415,335, Paper 9, Response to Office Action, EPIC000489-501
23.	Excerpt from Deposition transcript of Paul C, Clark, July 11, 2008
24.	Excerpt from Deposition transcript of Keith Lowery, April 25, 2008
25.	Excerpt from Deposition transcript of Katrina Montinola, February 12, 2008
26.	Excerpt from Deposition transcript of Matthew Sarboraria, May 15, 2008
27.	Excerpt from Deposition transcript of Michael Shamos, June 30, 2008
28.	Excerpt from Deposition transcript of Douglas Surber, February 28, 2008
29.	Oracle WebServer User's Guide Release 2.0 Production, Part No. A23646-1 "WebServer 2.0 Guide", ORCL01816630-01816882

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, David E. Moore, hereby certify that on August 7, 2008, the attached document was electronically filed with the Clerk of the Court using CM/ECF which will send notification to the registered attorney(s) of record that the document has been filed and is available for viewing and downloading.

I further certify that on August 7, 2008, the attached document was Electronically Mailed to the following person(s):

Mary B. Graham
Morris, Nichols, Arsht & Tunnell LLP
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
MGraham@MNAT.com

James G. Gilliland
Igor Shoiket
Townsend and Townsend and Crew LLP
Two Embarcadero Center
Eighth Floor
San Francisco, CA 94111-3834
OracleEpicrealm@townsend.com

Theodore T. Herhold
Robert J. Artuz
Eric M. Hutchins
Eric A. Mercer
Joseph A. Greco
Nitin Gupta
Townsend and Townsend and Crew LLP
379 Lytton Avenue
Palo Alto, CA 94301
OracleEpicrealm@townsend.com

Chad E. King
Townsend and Townsend and Crew LLP
1200 Seventeenth Street
Suite 2700
Denver, CO 80202
OracleEpicrealm@townsend.com

/s/ David E. Moore
Richard L. Horwitz
David E. Moore
Potter Anderson & Corroon LLP
Hercules Plaza – Sixth Floor
1313 North Market Street
P.O. Box 951
Wilmington, DE 19899-0951
(302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

Exhibit 1

US005894554A

United States Patent [19]

Lowery et al.

[11] Patent Number: 5,894,554

[45] Date of Patent: Apr. 13, 1999

[54] **SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS**

[75] Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of Tex.

[73] Assignee: **InfoSpinner, Inc.**, Richardson, Tex.

[21] Appl. No.: **08/636,477**

[22] Filed: **Apr. 23, 1996**

[51] **Int. Cl.**[6] **G06F 13/14**; G06F 13/20

[52] **U.S. Cl.** **395/200.33**; 395/200.68; 395/200.75; 395/680; 707/10; 707/104

[58] **Field of Search** 358/400; 395/800, 395/700, 200.68, 200.75, 200.53, 680, 200.33; 707/104, 10

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,866,706	9/1989	Christophersen et al.	370/85.7
5,341,499	8/1994	Doragh	395/700
5,392,400	2/1995	Berkowitz et al.	395/200
5,404,522	4/1995	Carmon et al.	395/650
5,404,523	4/1995	DellaFera et al.	395/650
5,404,527	4/1995	Irwin et al.	395/700
5,452,460	9/1995	Distelberg et al.	395/700
5,532,838	7/1996	Barbari	358/400
5,751,956	5/1998	Kirsch	395/200.33
5,761,673	6/1998	Bookman et al.	707/104

OTHER PUBLICATIONS

"Beyond the Web: Excavating the Real World Via Mosaic", Goldberg et al. Second International WWW, Oct. 17, 1994.

PCT International Search Report, Aug. 21, 1997.

Primary Examiner—Thomas C. Lee

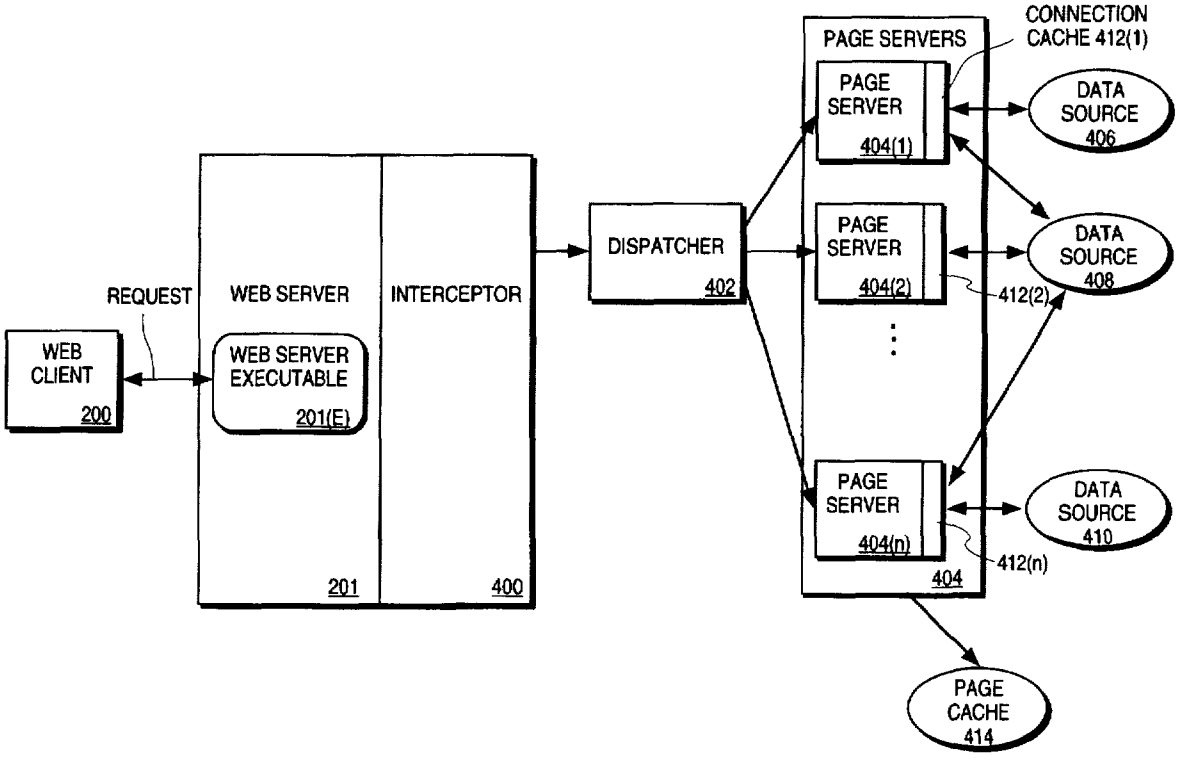
Assistant Examiner—Rehana Perveen

Attorney, Agent, or Firm—Blakely, Sokoloff, Taylor & Zafman LLP

[57] **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

11 Claims, 5 Drawing Sheets



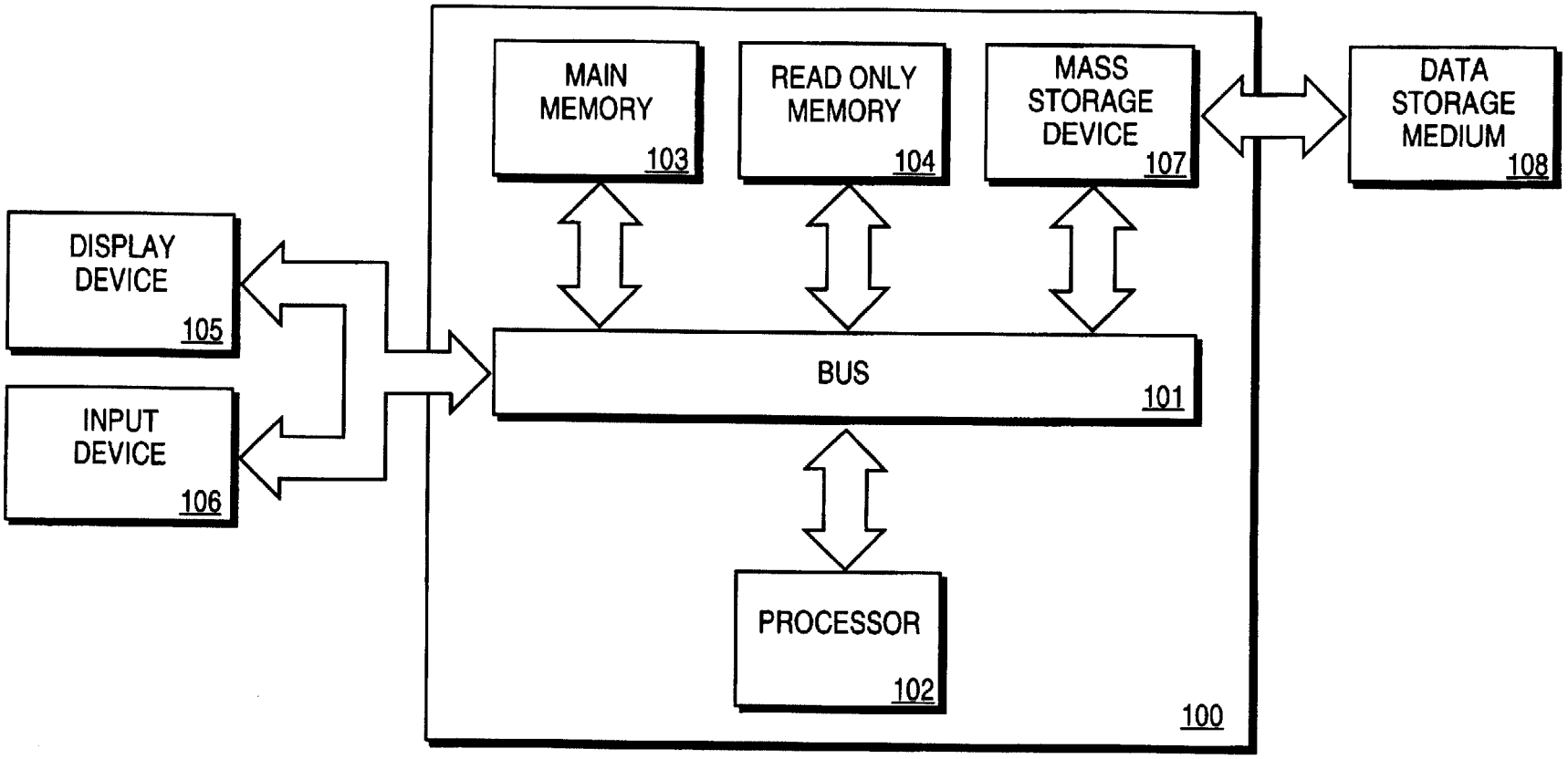

MAIN MEMORY
103
READ ONLY MEMORY
104
MASS STORAGE DEVICE
107
DATA STORAGE MEDIUM
108
DISPLAY DEVICE
105
INPUT DEVICE
106
BUS
101
PROCESSOR
102
100

FIG. 1

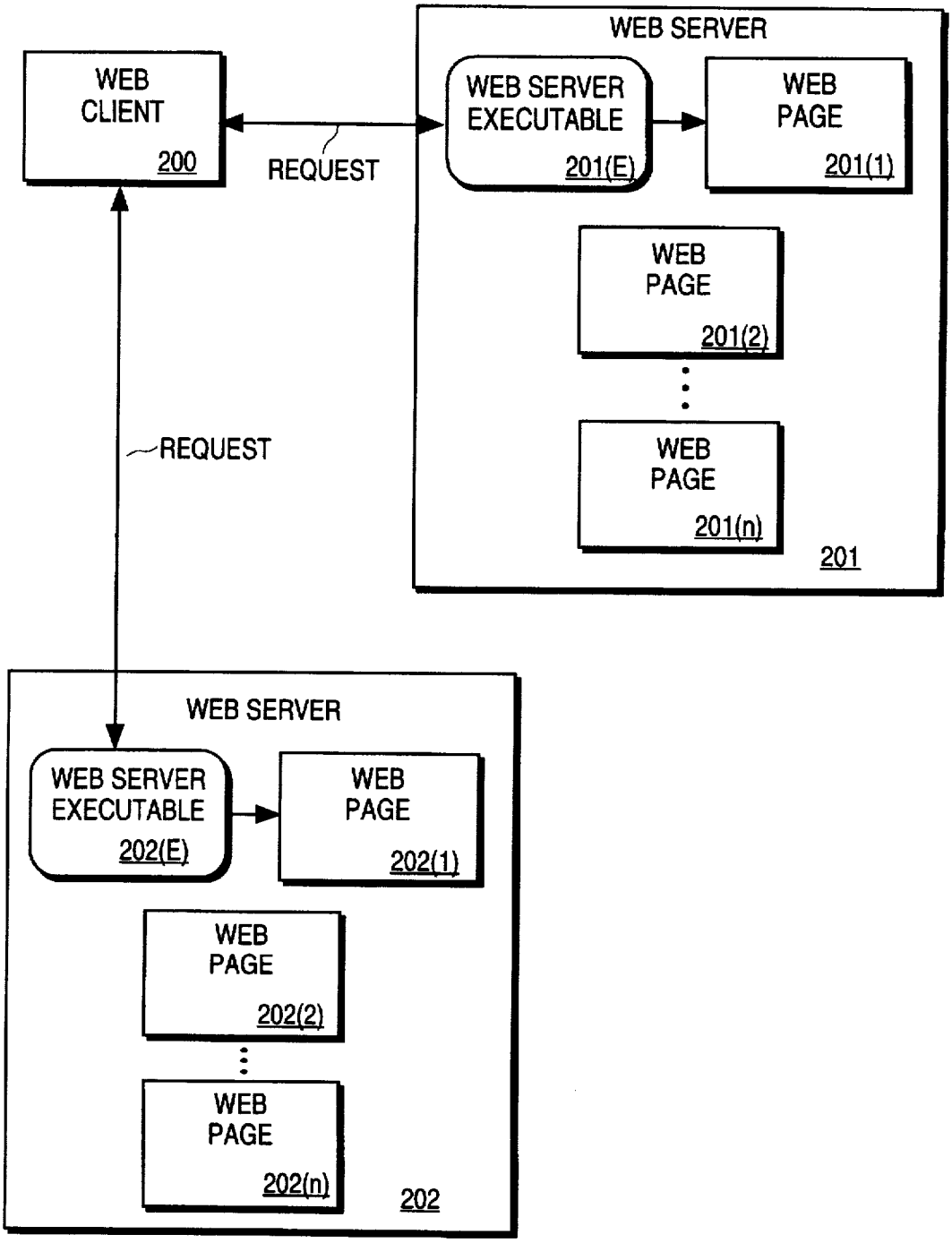


FIG. 2 (PRIOR ART)

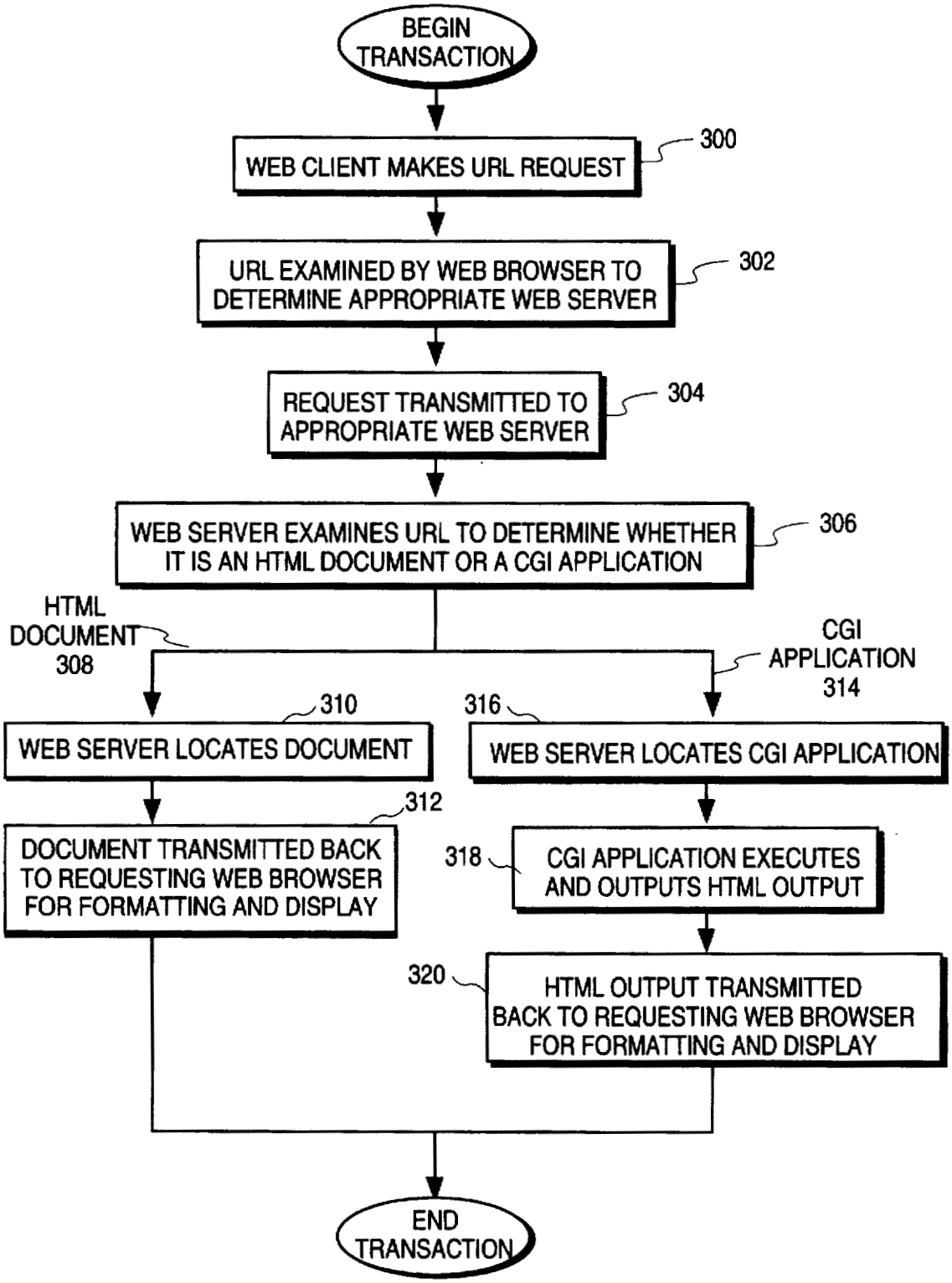


FIG. 3 (PRIOR ART)

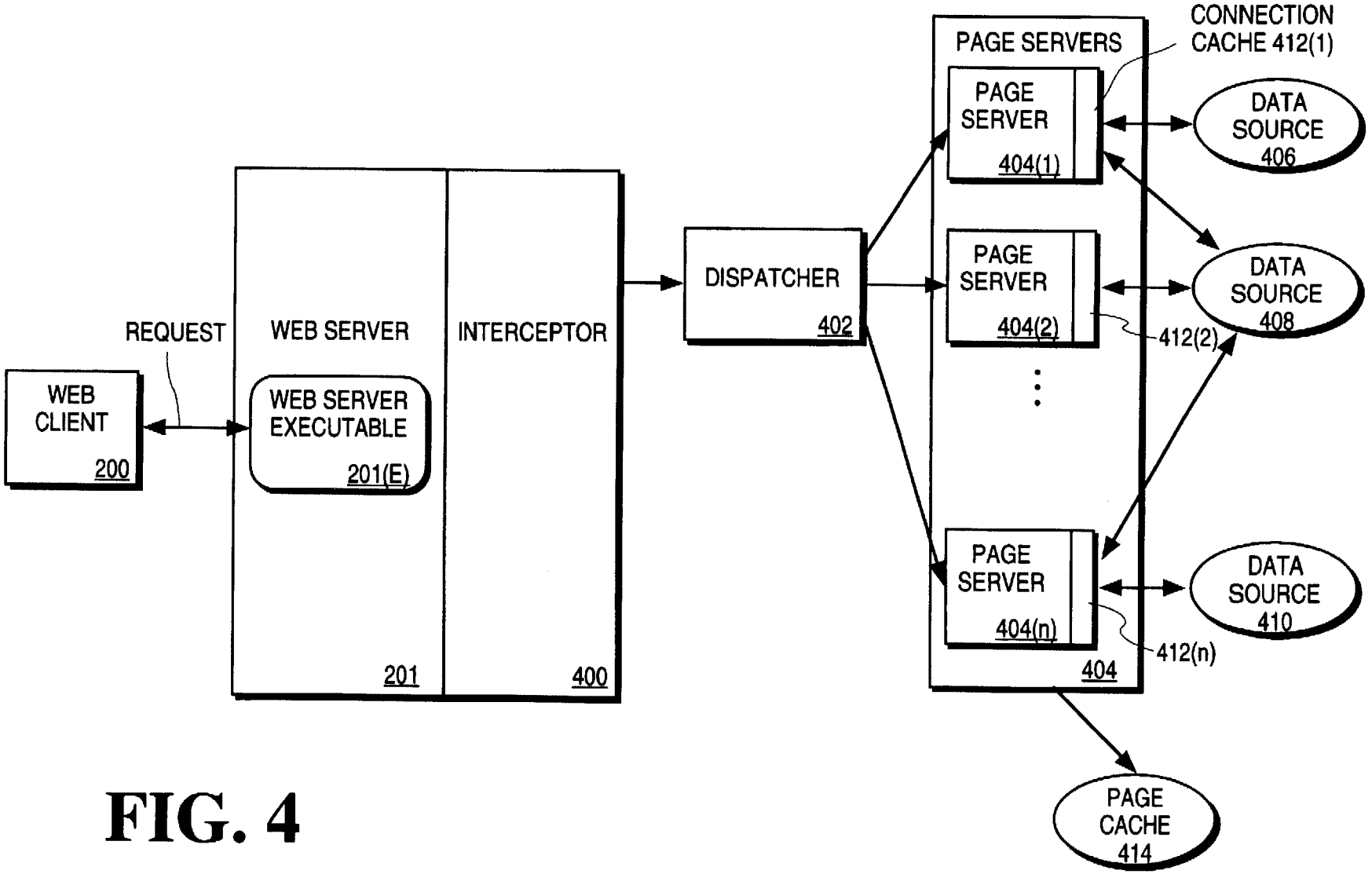

WEB CLIENT
200
REQUEST
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
INTERCEPTOR
400
DISPATCHER
402
PAGE SERVERS
CONNECTION CACHE 412(1)
PAGE SERVER
404(1)
DATA SOURCE
406
PAGE SERVER
404(2)
412(2)
DATA SOURCE
408
PAGE SERVER
404(n)
412(n)
404
DATA SOURCE
410
PAGE CACHE
414

FIG. 4

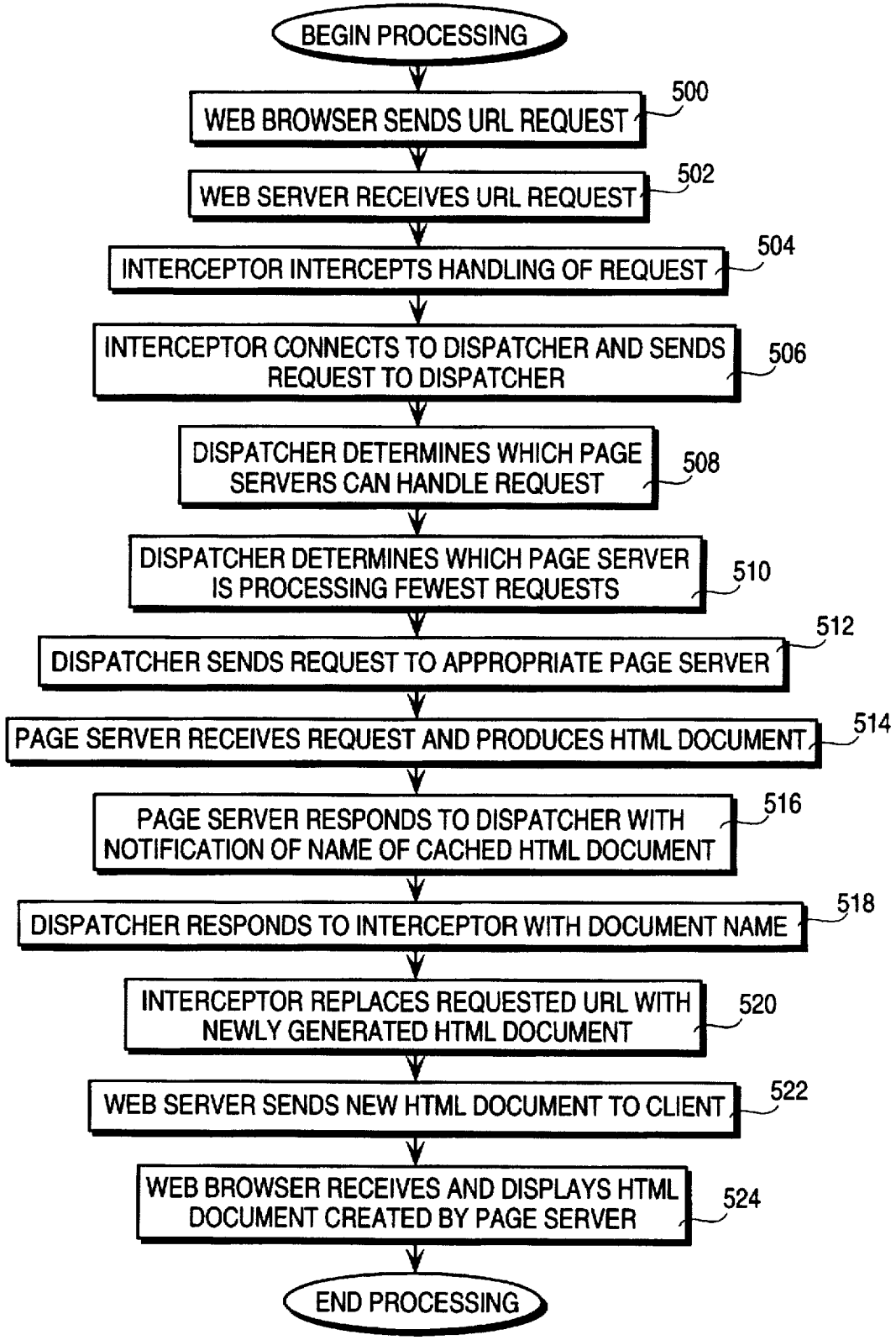

BEGIN PROCESSING
WEB BROWSER SENDS URL REQUEST
500
WEB SERVER RECEIVES URL REQUEST
502
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
504
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
506
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
508
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
510
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
512
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
514
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
516
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
518
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
520
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
522
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
524
END PROCESSING

FIG. 5

SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the farm of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, N.Y. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, N.Y. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. 1 comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404** (1)–(n). For example, if Page server **404** (1) receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404** (1) can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404**(1)–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404**(3), for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404**(3). This "connection caching" functionality is described in more detail below, under the heading "Performance."

Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404**(1) and **404**(2) are both logged into Data source **408**, but that Page server **404**(2) has the financial information already cached in Page server **404**(2)'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404**(2) to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404**(1)–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404**(2), receives the request, Page server **404**(2) will process the request. While Page server **404**(2) is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404**(2) and Web server executable **201**(E) to simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404**(2) dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability.

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404**(1) can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404**(1) can maintain connection cache **412**(1), containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize

the performance of the data source being utilized. This "page caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412**(1)–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging gent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access o the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing said request from said Web server to a page server, said page server receiving said request and releasing

said Web server to process other requests, wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

3. The computer-implemented method in claim **2** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

4. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

5. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of logging into said one or more data sources.

6. The computer-implemented method in claim **3** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

7. The computer-implemented method in claim **3** wherein said page server includes custom HTML extension templates for configuring said Web page.

8. The computer-implemented method in claim **7** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

9. A networked system for managing a dynamic Web page generation request, said system comprising:

one or more data sources;

a page server having a processing means;

a first computer system including means for generating said request; and

a second computer system including means for receiving said request from said first computer, said second computer system also including a router, said router routing said request from said second computer system to said page server, wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server said page server receiving said request and releasing said second computer system to process other requests, said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

10. The networked system in claim **9** wherein said router in said second computer system includes:

an interceptor intercepting said request at said second computer system and routing said request; and

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

11. A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of:

routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page, said Web page including data retrieved from one or more data sources.

* * * * *

Exhibit 2

US006415335B1

(12) **United States Patent**
Lowery et al.

(10) **Patent No.: US 6,415,335 B1**
(45) **Date of Patent: *Jul. 2, 2002**

(54) **SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS**

(75) Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of TX (US)

(73) Assignee: **epicRealm Operating Inc.**, Richardson, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/234,048**

(22) Filed: **Jan. 19, 1999**

Related U.S. Application Data

(62) Division of application No. 08/636,477, filed on Apr. 23, 1996, now Pat. No. 5,894,554.

(51) **Int. Cl.**[7] **G06F 13/14**; G06F 13/20
(52) **U.S. Cl.** **710/5**; 710/7; 709/219; 709/223; 709/238
(58) **Field of Search** 709/238, 223, 709/219; 710/5, 7, 20–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,866,706	A		9/1989	Christophersen et al. .. 370/85.7
5,341,499	A		8/1994	Doragh 395/700
5,392,400	A		2/1995	Berkowitz et al. 395/200
5,404,522	A		4/1995	Carmon et al. 395/650
5,404,523	A		4/1995	DellaFera et al. 395/650
5,404,527	A	*	4/1995	Irwin et al. 395/700
5,452,460	A		9/1995	Distelberg et al. 395/700
5,532,838	A		7/1996	Barbari 358/400
5,701,463	A	*	12/1997	Malcolm 395/610
5,751,956	A		5/1998	Kirsch 395/200.33
5,752,246	A	*	5/1998	Rogers et al. 707/10
5,754,772	A	*	5/1998	Leaf 395/200.33
5,761,673	A	*	6/1998	Bookman et al. 707/104
5,774,660	A		6/1998	Brendel et al. 395/200.31
5,774,668	A		6/1998	Choquier et al. 395/200.53

OTHER PUBLICATIONS

Hoffner, 'Inter–operability and distributed application platform design', Web URL:http://www.ansa.co.uk/, 1995, pp. 342–356.*

Mourad et al, 'Scalable Web Server Architectures', IEEE, Jun. 1997, pp. 12–16.*

Dias et al, 'A Scalable and Highly Available Web Server', IEEE, 1996, pp. 85–92.*

'Single System Image and Load Balancing for Network Access to a Loosely Coupled Complex', IBM TDB, vol. 34, Feb. 1992, pp. 464–467.*

Dias, Daniel M., et al.; A Scalable and Highly Available Web Server; IBM Research Division; T.J. Watson Research Center; 7 pages.

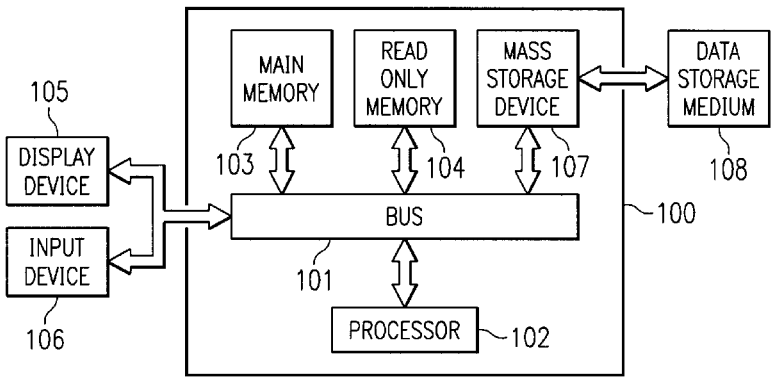
(List continued on next page.)

Primary Examiner—Jeffrey Gaffin
Assistant Examiner—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Baker Botts L.L.P.

(57) **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

29 Claims, 4 Drawing Sheets



OTHER PUBLICATIONS

Andresen, Daniel, Et Al.; Scalability Issues for High Performance Digital Libraries on the World Wide Web; Department of Computer Science; University of California at Santa Barbara; 10 pages.

Andresen, Daniel, Et Al.; SWEB: Towards a Scalable World Wide Web Server on Multicomputers; Department of Computer Science; University of California at Santa Barbara; 7 pages.

Holmedahl, Vegard; Et Al.; Cooperative Caching of Dynamic Content on a Distributed Web Server; Department of Computer Science; University of California at Santa Barbara; 8 pages.

Overson, Nicole; NeXT Ships WebObjects—On Time—As Promised; Deja.com: NeXT Ships WebObjects—On Time—As Promishttp://X28..deja.com/=dnc/ST_m=ps...EXT=927585438. 1744765032&hitnum=33.

International Search Report; 7 pages; dated Aug. 21, 1997.

Birman, Kenneth P. and van Renesse, Robbert; Software for Reliable Networks; Scientific American; May 1996; pp. 64–69.

"Beyond the Web: Excavating the Real World Via Mosaic"; Goldberg et al.; Second International WWW Conference; Oct. 17, 1994.

* cited by examiner

FIG. 1

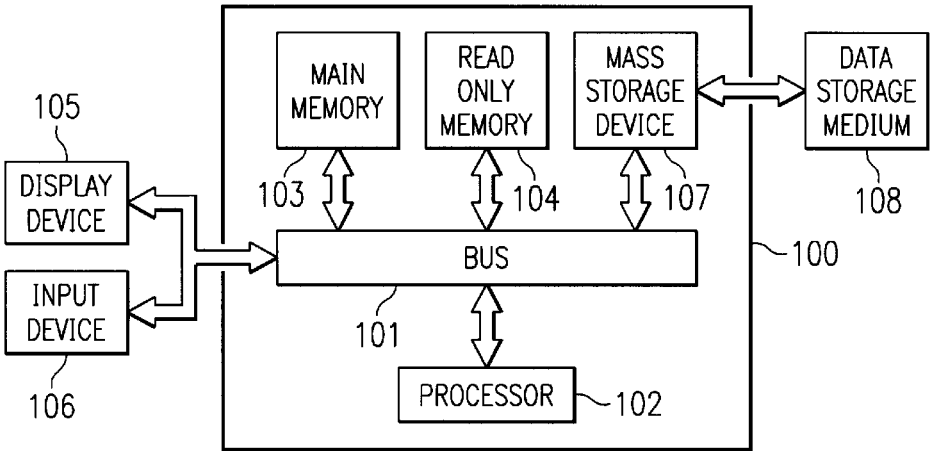


FIG. 2
(PRIOR ART)

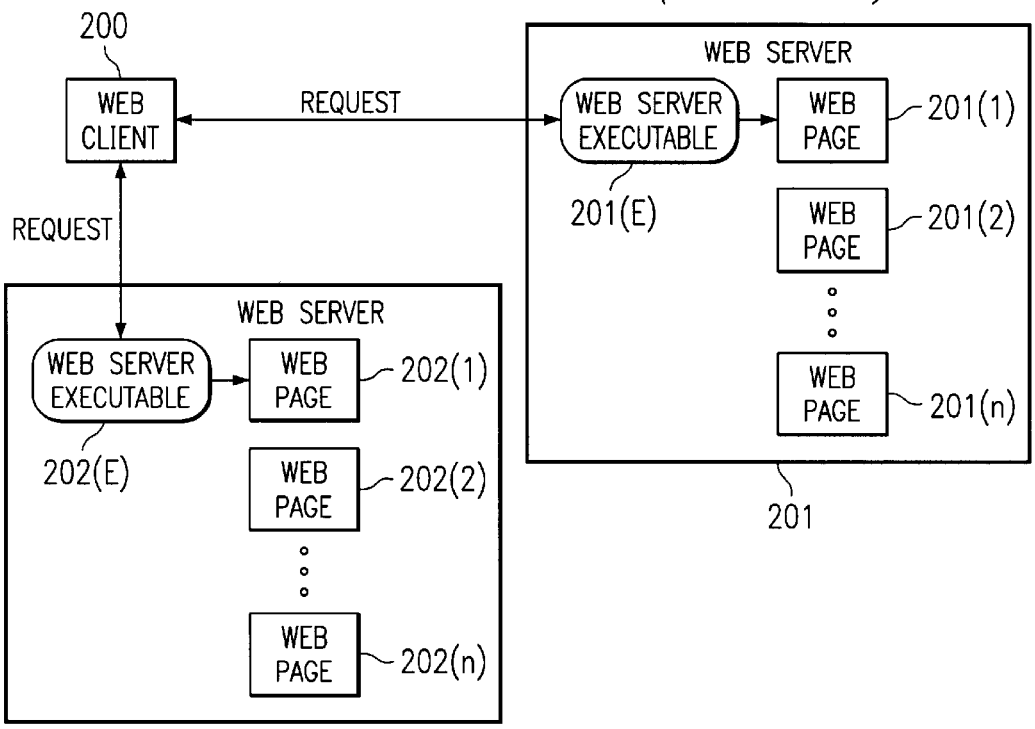


FIG. 3
(PRIOR ART)

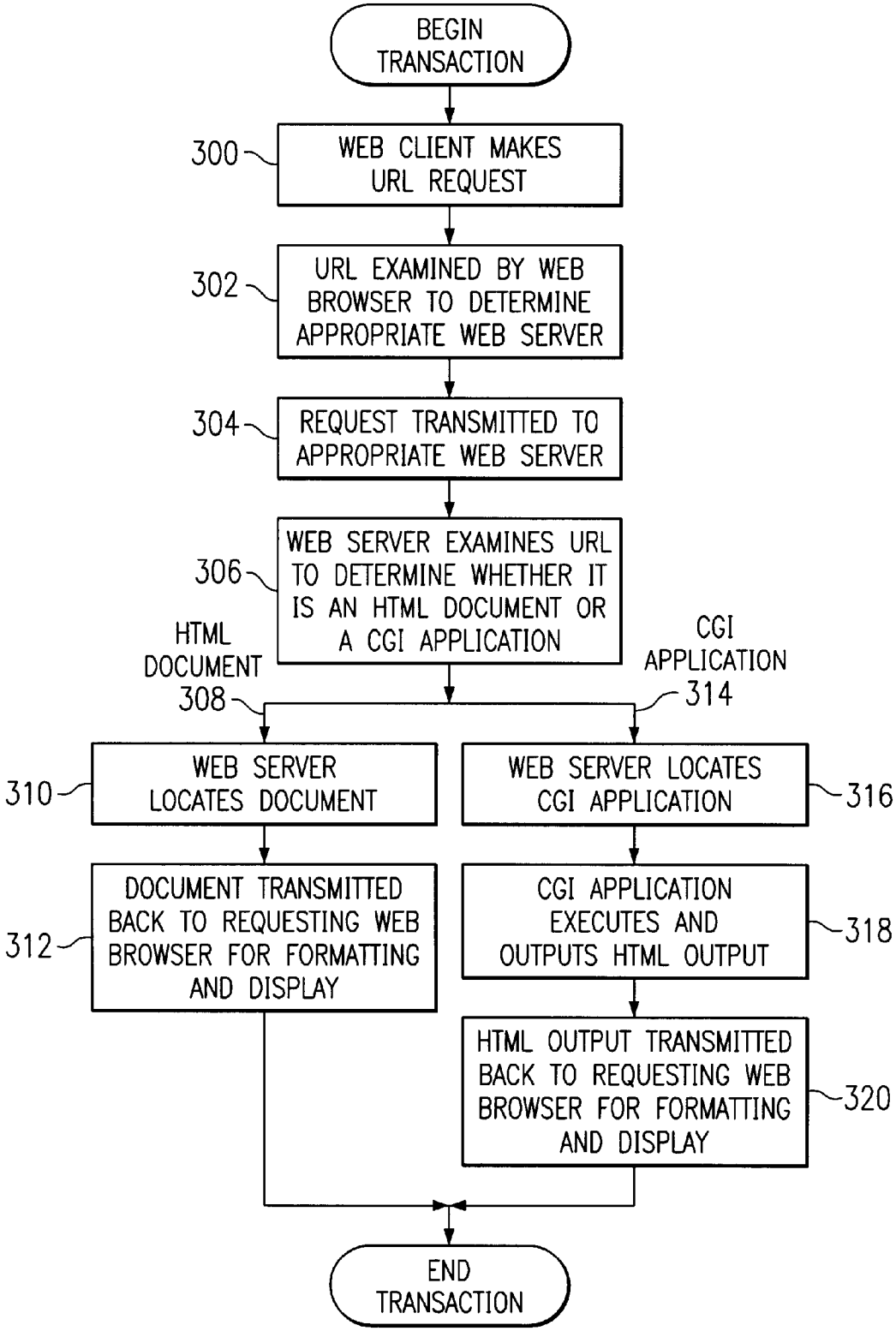


FIG. 4

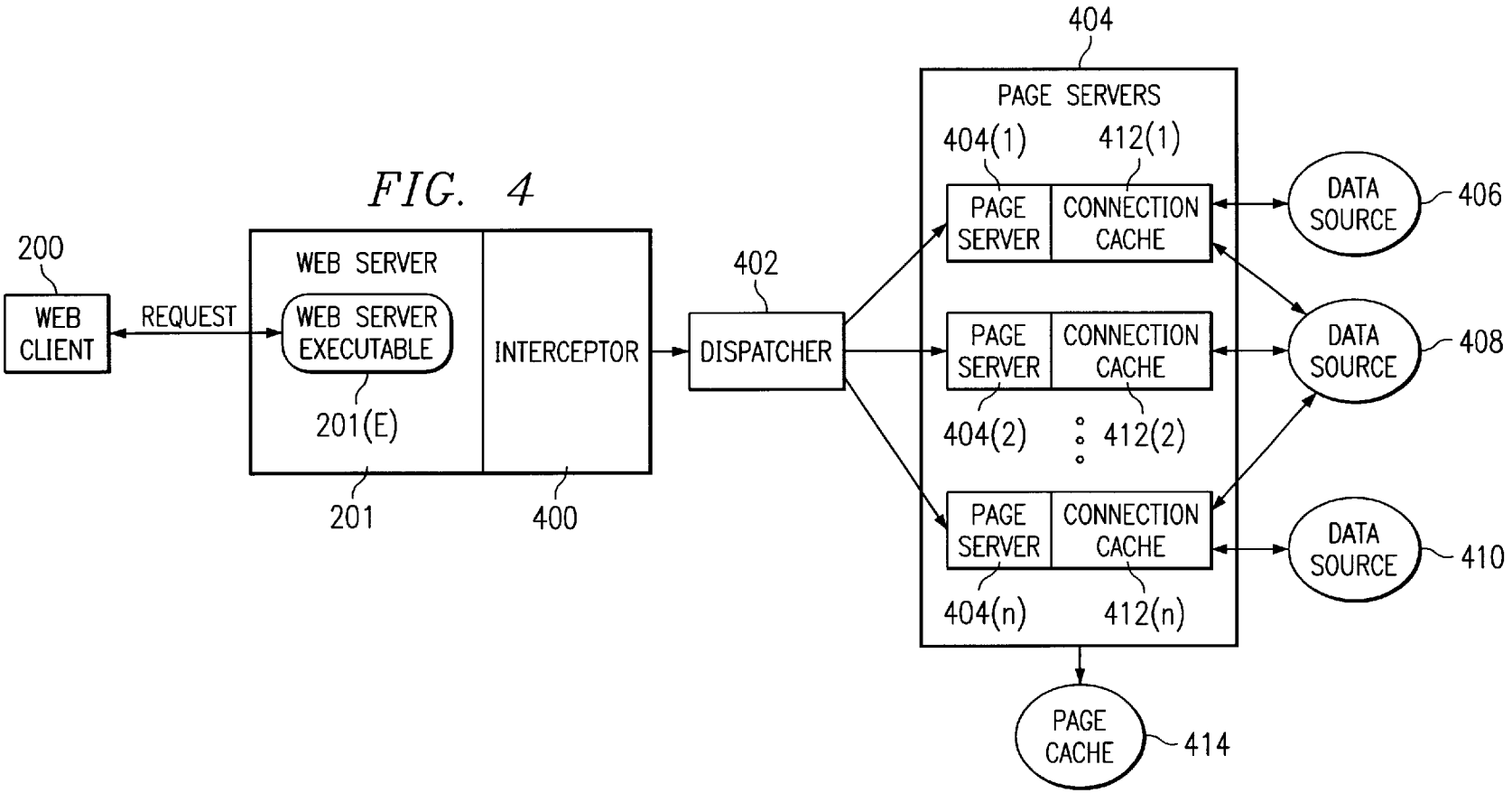
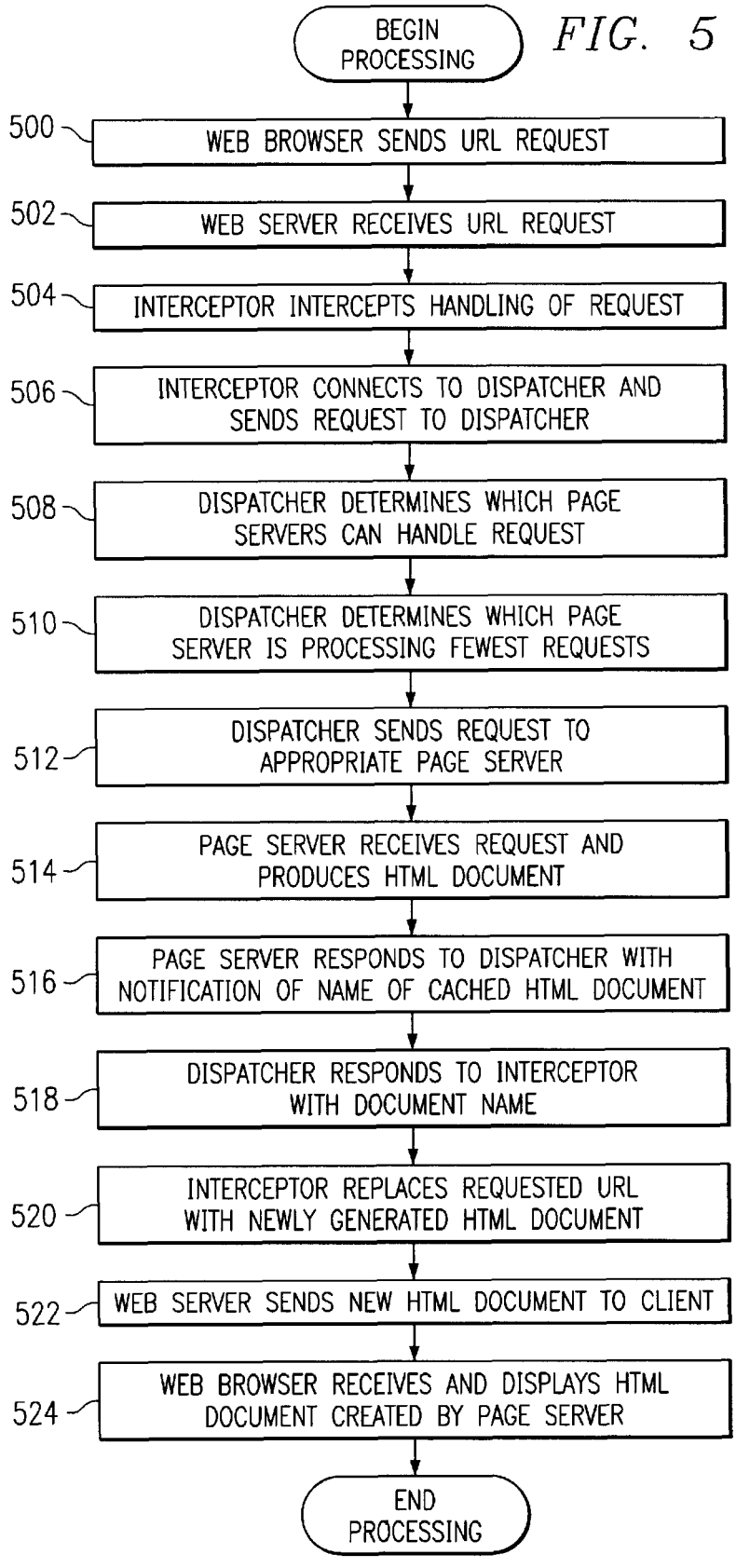

200
WEB CLIENT
REQUEST
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
INTERCEPTOR
400
402
DISPATCHER
404
PAGE SERVERS
404(1)
412(1)
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
406
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
408
404(2)
412(2)
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
410
404(n)
412(n)
PAGE CACHE
414

FIG. 5


BEGIN PROCESSING
500
WEB BROWSER SENDS URL REQUEST
502
WEB SERVER RECEIVES URL REQUEST
504
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
506
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
508
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
510
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
512
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
514
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
516
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
518
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
520
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
522
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
524
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
END PROCESSING

SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS

This application is a division of Ser. No. 08/636,477, filed Apr. 23, 1996, now U.S. Pat. No. 5,894,554.

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, New York. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, New York. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. **1** comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404 (1)**–(n). For example, if Page server **404 (1)** receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404 (1)** can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404(1)**–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404(3)**, for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404(3)**. This "connection caching" functionality is described in more detail below, under the heading "Performance." Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404(1)** and **404(2)** are both logged into Data source **408**, but that Page server **404(2)** has the financial information already cached in Page server **404(2)**'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404(2)** to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404(1)**–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404(2)**, receives the request, Page server **404(2)** will process the request. While Page server **404(2)** is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404(2)** and Web server executable **201**(E) to. simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404(2)** dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404(1)** can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404(1)** can maintain connection cache **412(1)**, containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize the performance of the data source being utilized. This "page

caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412(1)**–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging agent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access to the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing a request from a Web server to a page server, said page server receiving said request and releasing said

Web server to process other requests wherein said routing step further includes the steps of:

intercepting said request at said Web server and routing said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim **1** wherein said step of routing said request includes the steps of:

routing said request from said Web server to a dispatcher; and

dispatching said request to said page server.

3. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

4. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

5. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

6. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of logging into said one or more data sources.

7. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

8. The computer-implemented method in claim **1** wherein said page server includes tag-based text templates for configuring said Web page.

9. The computer-implemented method in claim **8** wherein said step of processing said request further includes the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates.

10. The computer-implemented method in claim **8** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

11. The computer-implemented method in claim **8** wherein said tag-based text templates include HTML templates.

12. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

13. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of processing an object handling extension.

14. The computer-implemented method in claim **13** wherein said object handling extension is an OLE extension.

15. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a page server, said page server receiving said request and releasing said HTTP-compliant device to process other requests wherein said transferring step further includes the steps of:

intercepting said request at said HTTP-compliant device and transferring said request to said page server;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

16. The computer-implemented method in claim **15** wherein said step of transferring said request includes the steps of:

transferring said request from said HTTP-compliant device to a dispatcher; and

dispatching said request to said page server.

17. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

18. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of dynamically retrieving said data from said one or more data sources.

19. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

20. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of logging into said one or more data sources.

21. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of maintaining a page cache containing said page.

22. The computer-implemented method in claim **15** wherein said page server includes tag-based text templates for configuring said page.

23. The computer-implemented method in claim **22** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates.

24. The computer-implemented method in claim **22** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

25. The computer-implemented method in claim **22** wherein said tag-based text templates include HTML templates.

26. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

27. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of processing an object handling extension.

28. The computer-implemented method in claim **27** wherein said object handling extension is an OLE extension.

29. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a dispatcher;

maintaining dynamic information regarding data sources a given page server may access;

dispatching said request to an appropriate page server based on said request and based on said dynamic information, said page server receiving said request and

releasing said HTTP-compliant device to process other requests;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

* * * * *

Exhibit 3

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC., **Plaintiffs,** **v.** **EPICREALM LICENSING, LLC,** **Defendant.**	**C.A. No. 06-cv-414 (SLR)**
AND RELATED COUNTERCLAIMS	

REPORT OF PLAINTIFFS' EXPERT MICHAEL I. SHAMOS, PH.D, J.D. CONCERNING INVALIDITY

BACKGROUND & QUALIFICATIONS

1. My name is Michael I. Shamos. I hold the title of Distinguished Career Professor in the School of Computer Science at Carnegie Mellon University in Pittsburgh, Pennsylvania. I was a founder and Co-Director of the Institute for eCommerce at Carnegie Mellon and I now direct a graduate degree program in eBusiness Technologies. My résumé is attached as Exhibit 2 to this report.

2. I currently teach graduate courses at Carnegie Mellon in Electronic Commerce, including eCommerce Technology, Electronic Payment Systems, Electronic Voting and eCommerce Law and Regulation and have done so since 1999. In Fall 2007 I taught a new course entitled Law of Computer Technology.

3. From 1979-1987 I was the founder and president of two computer software development companies in Pittsburgh, Pennsylvania, Unilogic, Ltd. and Lexeme Corporation.

4. I am an attorney admitted to practice in Pennsylvania and have been admitted to the Bar of the U.S. Patent and Trademark Office since 1981. Despite being an attorney, I have not been asked to offer any opinions on patent law in this action.

5. I have previously testified in a number of cases concerning computer software, intellectual property, electronic payment, electronic voting and electronic commerce matters. My résumé in Exhibit 2 contains a list of cases in which I have testified in the last ten years.

6. I have been retained as a technical expert by Townsend, Townsend & Crew LLP, counsel for Oracle Corporation and Oracle U.S.A. Inc. (collectively, "Oracle") in this action.

7. I have been engaged through Expert Engagements LLC ("EE"), a firm that locates expert services for law firms. EE charges $475 per hour for my services, of which I receive $425. I am one of the owners of EE. No part of my compensation is dependent on the outcome of this case.

8. I have been asked by counsel for Oracle to offer an expert opinion on the validity of claims 1-11 of Lowery et al. U.S. Patent 5,894,554 (the '554 Patent) and claims 2, 6 and 29 of Lowery et al. U.S. Patent 6,415,335 (the '335 Patent), collectively the "Asserted Claims" of the "Patents" based on the claim constructions propounded by Oracle and epicRealm. I note that many of these constructions are identical or similar to Markman Constructions in the Texas Actions as contained in the "Report and Recommendation Regarding Claim Construction," dated August 15, 2006 (the "Markman Report"), as adopted by the Court's Order Adopting Report and Recommendations Concerning Claim Construction" dated October 30, 2006 (the "Markman Order"). I will refer to the constructions in the Texas Actions as the "Markman Constructions" even though I am aware that this Court had not made such constructions yet in the present case. The Asserted Claims share a common specification (the "Specification"). The fact, however, that I rely on the Texas Court's claim constructions or epicRealm's proposed claim constructions should not suggest that I necessarily I agree with those constructions or positions. Where I agree

or disagree with certain constructions or positions, I have expressly stated so. In connection with my analysis, I have reviewed the Patents, their prosecution histories and the documents listed in Exhibit 1, attached hereto.

9. I have also been engaged by defendants in three cases in the Eastern District of Texas captioned (1) epicRealm Licensing, LLP v. Autoflex Leasing, Inc. at al., Civil Action No 5:07-CV-125; (2) epicRealm Licensing, LLP v. Franklin Covey Co. et al., Civil Action No 5:07-CV-126; and (3) epicRealm Licensing, LLP v. Various, Inc., Civil Action No 5:07-CV-135 (collectively, the "Texas Actions"). The Texas Actions assert that the various defendants are infringing all of the claims of the Patents. The issues with respect to validity of the Patents in the present action are essentially identical to those in the Texas Actions. I have submitted an expert report on invalidity in those cases that is substantially similar to this one with respect to the claims that have been asserted against defendants in the Texas Actions. To avoid confusion, I will refer to the defendants in the Texas Actions as the "Texas Defendants."

10. The individuals named as inventors in the Patents are referred to collectively herein as the "Applicants."

11. In this report, where I have cited a reference as prior art, either the reference predates the filing date of the Patent or I have been informed by counsel for Oracle that Oracle will be able to prove at trial that the reference is prior art as to the Patent. I have not made any independent determination of relative dates of invention or publication, except that I express opinions herein concerning the evidence epicRealm has offered in an attempt to prove dates of conception and reduction to practice.

12. It may be necessary for me to revise or supplement this report based on material subsequently presented by epicRealm, and I reserve the right to do so. I may also present demonstrative evidence at trial, and I reserve the right to do so. In particular, I am informed that depositions have recently been taken of inventors Howell, Levine, and Lowery and of a third party witness named J.J. Allaire. Additionally, I am informed that a deposition of IBM is scheduled to take place in May regarding potentially invalidating prior art. I am also informed

that epicRealm has recently supplemented its infringement contentions to accuse an Oracle Database product called "RAC" and other database components including JDBC and OCI. I may need to revise or supplement this report based on recent deposition testimony or newly asserted infringement theories because the timing of these events did not provide me with sufficient time to consider or fully respond to such new discovery.

13. It may be necessary for me to revise or supplement this report, or file a supplemental or responsive report, based on any responsive report of epicRealm, and I reserve the right to do so.

14. I am aware that the patents-in-suit are now the subject of reexamination proceedings before the U.S. Patent and Trademark Office. The '554 Reexamination Requests and the '335 Reexamination Requests will be referred to herein as the "Reexamination Requests."

15. My opinions are also based on epicRealm's filings and other papers in both the Texas and Delaware actions.

SUMMARY OF OPINIONS

16. This section is not intended to be a complete list of the opinions I have reached.

17. Each Asserted Claim is anticipated by at least 12 prior art references, and in some cases by 30 references. Each independent claim of the Patents is anticipated by at least 29 references.

18. Every Asserted Claim is obvious in light of numerous combinations of prior art references in which the suggestion to combine, to the extent one is required, is clear from the references themselves and the development of the art as discussed below.

19. All of the Asserted Claims are invalid under 35 U.S.C. §112.

20. None of the Asserted Claims is valid.

21. The documents relied upon by epicRealm to demonstrate conception prior to the filing date of the '554 patent (EPIC000001-173) fail to do so.

22. I express no opinion on the issue whether the claimed inventions were in use or on sale by epicRealm or its predecessors so as to act as a statutory bar because I am not in possession of facts sufficient to make such a determination.

23. I am in agreement with the conclusions drawn by an outside consultant, Dr. Shuang Chen, in "A Special Report on Analysis of Prior Arts to US Patent 5,894,554," dated December 11, 2003, which was communicated to epicRealm prior to the filing of the present action. Dr. Chen was asked by an attorney to consider the enforceability of the '554 patent. He concluded as follows: "The prior arts demonstrated that the method of server redirection for web page generation had existed prior to April 1996 when the US Patent 5,894,554 was filed, and the prior arts are very closely related to the said patent. Further, the prior arts for dynamic web page generation, a network system and machine-readable medium for storing computer instruction data could easily be established. Given the long time existence of the DNS technology prior to the said patent and the massive development efforts of the World Wide Web standards in public domain, any attempted argument into the subtle differences between the method described in this patent and the prior arts would be difficult to pass the 'obviousness' test, and be used for the purpose of enforcing this patent. Therefore, this report concludes that the US Patent (5,894,554) would hardly be enforceable."

LEGAL PRINCIPLES

24. I have not been asked to offer an opinion on the law; however, as an expert assisting the Court in determining claim validity, I understand that I am obliged to follow existing law. I have therefore been asked to apply the following legal principles to my analysis of allegations of invalidity in light of the prior art. I will discuss the legal standards I have been asked to apply concerning allegations of invalidity pursuant to 35 U.S.C. §112 in the appropriate section later in this report.

a. For a claim to be anticipated, every limitation of the claimed invention must be found in a single prior art reference, either expressly or inherently, arranged as in the claim.

b. When a claim covers several alternative structures or compositions of elements, either generically or as alternatives, the claim is deemed anticipated if any of the structures or compositions within the scope of the claim is disclosed or practiced in the prior art.

c. For a claim element to be inherently present in a prior art reference, the element must be "necessarily present" in the disclosed apparatus, system or method, not merely probably or possibly present.

d. A claim is invalid for obviousness if differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

e. In determining whether a claimed invention is obvious, one should consider the scope and content of the prior art, the level of ordinary skill in the relevant art, the differences between the claimed invention and the prior art, and whether the claimed invention would have been obvious to one of ordinary skill in the art in light of those differences.

f. I understand that certain objective factors, sometimes known as "secondary considerations" may also be taken into account in determining whether a claimed invention would have been obvious. One of these factors is contemporaneous development of the claimed invention by others. I may discuss that factor in this report, as it weighs in favor of a finding of obviousness. I have reviewed epicRealm's Response to Interrogatory No. 12 regarding "secondary considerations" and do not believe that such alleged considerations weigh in favor of nonobviousness, especially in view of the numerous prior art references discussed herein that anticipate the Asserted Claims and the numerous individuals and companies, including Oracle, discussed or referenced herein that developed the claimed inventions of the epicRealm patents prior to their filing dates.

To the extent that epicRealm seeks to raise other secondary considerations, I reserve the right to supplement my report to provide a rebuttal.

g. The person of ordinary skill is a hypothetical person who is presumed to be aware of all of the pertinent art. The person of ordinary skill is not an automaton, and may be able to fit together the teachings of multiple patents employing ordinary creativity and the common sense that familiar items may have obvious uses beyond their primary purposes. It is not necessary to demonstrate precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ. A patent which merely claims predictable uses of old elements according to their established functions to achieve predicable results should be found invalid as obvious.

h. In establishing obviousness, one must avoid the temptation to read into the prior art the teachings of the invention in issue and guard against slipping into the use of hindsight.

i. An invention is obvious if a designer of ordinary skill in the art, facing the wide range of needs created by developments in the field, would have seen an obvious benefit to the solutions tried by the applicant.

THE SUBJECT MATTER

25. The specifications of the Patents are substantially identical and therefore it makes sense to treat them collectively when describing their subject matter.

26. After the Internet was made available for general commercial use in the mid 1990s, the World Wide Web expanded prodigiously. Web browsing consists primarily in the viewing of Web pages, which are transmitted over the Internet from Web servers to individual machines called Web clients which run Web browsers. The principal function of a Web browser is to display, or "render" Web pages so they can be viewed by humans on display screens.

27. In the mid-1990s, most Web pages consisted of relatively fixed content that was not modified very frequently, e.g. less often than once per day. Such pages were referred to as "static." It made sense to store static pages on disk drives or in random-access memory so they could be delivered rapidly to Web clients that requested them. A typical example of a static page is the home page of a company giving various details such as the company's address and telephone number, along with link to other static pages that provide information about the

company's products. Static pages can be delivered quickly because virtually no processing is required to transmit them over a communication line.

28. Static pages, of course, are not suitable for information which changes frequently, such as the price of a stock, or for information that is retrieved or computed on a custom basis at the moment it is requested. Such pages are called "dynamic pages" because they are created dynamically on request. As a general matter, dynamic Web pages by their nature cannot be created or stored in advance[1]. A good example of a dynamic page is an individual user's shopping cart at a retail website. Obviously the shopping cart cannot be prestored because it is created dynamically in response to the shopping activity of the user. Another example of a dynamic page is one produced by a search engine such as Google, Lycos or Yahoo! in response to a query. Not only is a search engine's index of web pages changing dynamically as pages are being added to the Web and indexed by the engine, but there is no possible way to anticipate every possible query that might be posed. Therefore, the search engine response pages cannot be static, but must be generated on demand.

29. Because dynamic pages are created when they are requested, more computer resource is needed to produce and deliver a dynamic page than for a static page. It may in fact take thousands of times longer to generate a dynamic page. While Web servers can be extremely efficient at delivering static pages, their performance becomes unacceptably slow if they are also burdened with the computational load needed for dynamic pages. In order to maintain acceptable response times, it was known to send dynamic page requests to some server other than the Web server, restricting the Web server to handling only static requests. Once the dynamic page has been generated, it essentially becomes a static page, and can be passed back to the Web server for delivery to the Web client.

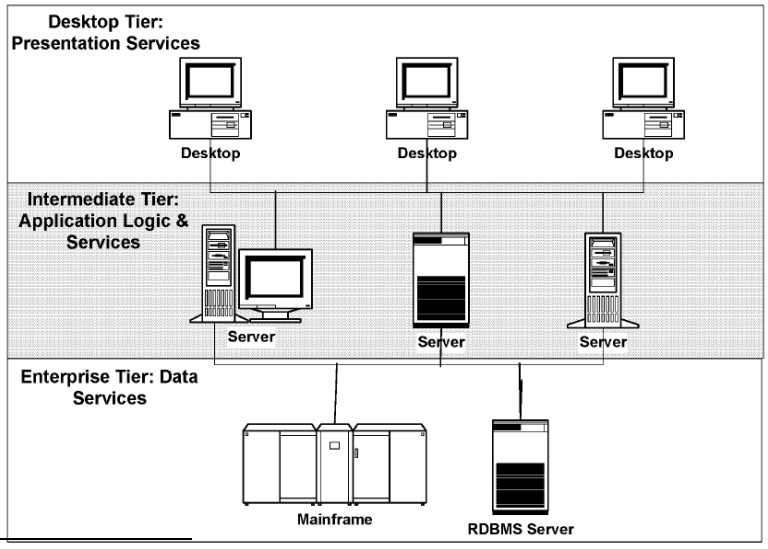
30. In this report, the phrase "it was known" means throughout that "it was known prior to the inventions claimed in the Patents."

[1] It is possible to store, or "cache" a dynamic page if there is an expectation that the very same page will be requested multiple times. Such caching would reduce the time necessary to generate the page repeatedly. However, it is not possible to store all possible dynamic pages that might be requested.

31. The Patents deal with methods for handling requests for dynamic pages by passing them off to one or more separate servers known as "page servers." To do this, the Patents make use of a well-known system architecture known as the "three-tier architecture," described in a paper by Herb Edelstein in the journal "DBMS" in May 1994 entitled, "Unraveling Client Server Architectures." In January 1995, Wayne W. Eckerson published a paper in the journal Open Information Systems entitled, "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications" (herein, Reference 193). A figure from that paper is reproduced below[2]. Tier 1 (called the "Desktop Tier" in the diagram) comprises the Web client, i.e. the computer on which a Web browser is running. Tier 2 (called the "Intermediate Tier" in the diagram) comprises the Web server. Tier 3 (the "Enterprise Tier") is shown as having a mainframe computer and a relational database management (RDMBS) server.

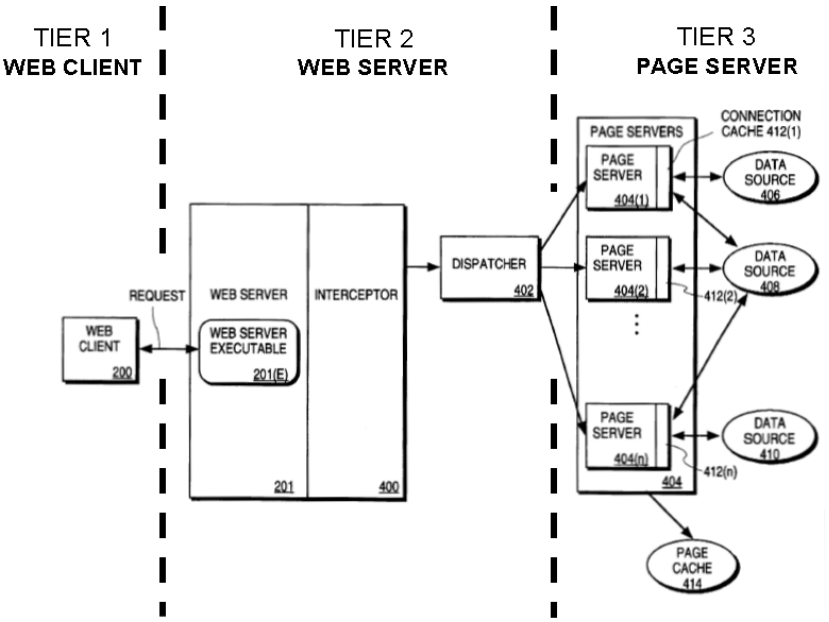
32. Static pages can be served through the Tier 2 servers (Web servers). Resource-intensive work, such as the generation of dynamic web pages, including retrieving dynamic data from databases and adding HTML markup, is performed by computers at Tier 3. The nature of the three-tier architecture is that the Tier 3 servers perform the heavy work of handling dynamic page requests, allowing the Tier 2 (Web) servers to continue rapid delivery of static pages.

Three-Tier Deployment Architecture



[2] The figure can be found at Bates page MAC003889.

33. That the Patents employ a standard three-tier architecture is apparent from Fig. 4 of the '554 Patent, reproduced below. The dotted vertical lines and the "Tier" headings have been added; otherwise, the drawing is unaltered.



34. While the diagram from the Eckerson reference shows the tiers drawn vertically, as a wedding cake, the figure from the Patents shows the tiers horizontally. However, there is no essential difference between the architectures.

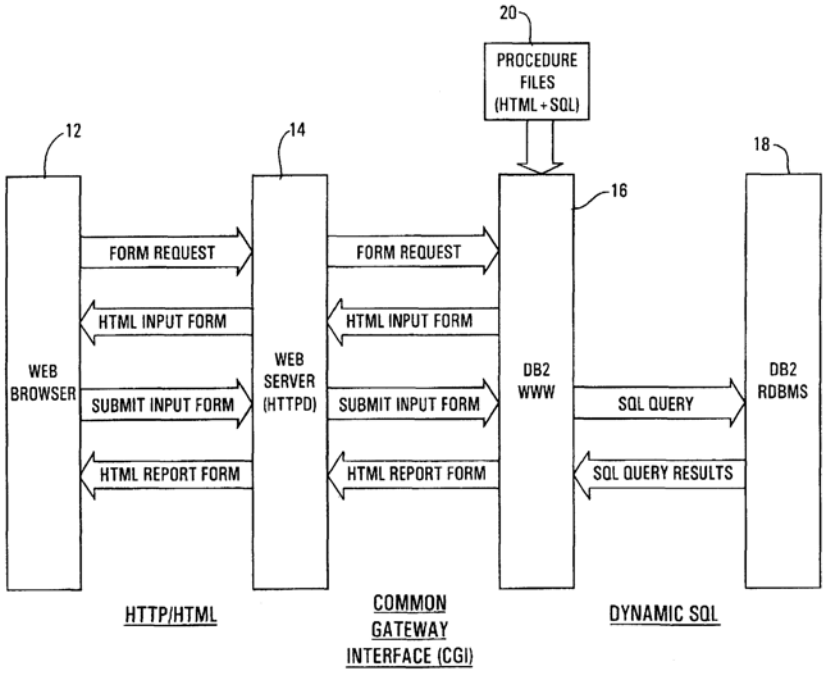
35. The figure from the Patents shows two boxes that are not expressly present in the Eckerson diagram, an "interceptor" and a "dispatcher." The function of the interceptor is to keep the Web server from spending time on dynamic page requests. Per my understanding of epicRealm's infringement theory for the "interceptor," it need only discriminate between requests that are to be handled by the Web server itself (static pages) and those that must be handed off to page servers (dynamic pages). Note that exactly the same function is performed in the three-tier architecture of Eckerson, even though no box specifically labeled "interceptor" is shown. This is because a decision must be made whether a request made by the Web client is to be handed by the Intermediate Tier (2) or the Enterprise Tier (3). This decision is made by code in the Web server which performed the function of intercepting.

36. Because Tier 3 in both Eckerson and the Patents comprises more than one server, some decision must be made which server should receive the request. Many methods of making such a decision are possible, including making a random choice. Eckerson, however, teaches making an intelligent choice based on a principle called "load balancing." Clearly, if multiple Tier 3 servers are available, but requests are sent to servers that are already busy or heavily loaded, the advantage of having multiple servers would be lost. It was known to keep track of how heavily loaded each server is, and to send the pending request to the least-busy server. Eckerson contains a section entitled, "Failover, Load-Balancing." It states, "The transaction monitor can balance fluctuating loads across multiple instances (servers) of an application resource. It can also sense when an application server has failed and automatically redirect requests to remaining available servers. This provides a high availability of system resources." MAC003894.

37. The intelligent choice of which Tier 3 server should receive the request is called "dispatching" in the Patents. The function of a dispatcher is performed in Eckerson by the code which implements the quoted failover and load-balancing logic in Tier 2. This is true even though there is no box expressly labeled "dispatcher" in the Eckerson diagram. There is no architectural difference between the prior art structure of Eckerson and Fig. 4 of the Patents[3].

38. The "data sources" accessed by Tier 3 of the Patents can sometimes be regarded as a fourth tier, resulting in a four-tier architecture. Four-tier architectures predated the Patents. Such an architecture for handling relational database queries over the Internet is disclosed in Reference 414, Ngyuen et al. U.S. Patent 5,737,592, "Accessing a relational database over the Internet using macro language files," having an application date of June 19, 1995. Fig. 2 of Nguyen et al., shown below, clearly indicates the four tiers.

[3] Fig. 4 of the Patents shows "connection cache 412(1)" and "page cache 414." These caches are the subject of several dependent claims and are discussed later, but are not fundamental features of the architecture.



39. Nguyen et al. discloses a fundamental structure of the Patents, namely the passing off of dynamic database queries to a separate server, interrogating the database, and substituting the results of the query into an HTML template, the result of which is returned to the user's web browser for display. The "page servers" and "data sources" of the Patents correspond to the "DB2 WWW" and "DB2 RDBMS" tiers in the Nguyen et al. drawing.

THE PRIOR ART

40. To determine what prior art is relevant, we must define the art. Both the '554 and the '335 Patents define the art as follows: "The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites." '554 1:9-11. The problem to be solved is further described later in the Patents: "Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites." '554 2:4-12.

41. The level of skill of one skilled in the art to which the Patents pertain would require familiarity with Internet programming, Web servers, methods of generating dynamic Web pages and familiarity with relevant Internet standards and protocols. No particular college degree would be required, nor would a person need to be a computer professional. Understanding the relevant technology presupposes programming skills but no advanced training.

42. The Patents address a problem that was well known at the time the applications were filed and for which many solutions were known, including the very matter claimed in the Patents. Internet use was increasing, and companies found a need for progressively more sophisticated web page content. The first web pages were static, that is, their content changed only very rarely, such as every few days or even less frequently. Therefore they could be constructed once and stored in their entirety on hard disk or even in random-access memory.

43. A web page is "requested" by a client computer and transmitted over the Internet by a server computer, usually just called a "server," or more precisely for purposes of this report, a "web server." The job of the web server is to receive requests for web pages and service those requests as rapidly as possible by transmitting the requested pages. Static pages can be accessed and delivered quickly by a web server. Any delay results in a slowdown, which degrades the performance of the website. Clients who are requesting pages will experience delays if the server is unable to handle the requested volume of requests. Therefore, it was well known in the art to "tune" servers to make them as fast as possible. One manner of tuning was to eliminate all application software on the web server that was not absolutely necessary for delivery of web pages. Another was to hold web pages in random-access memory rather than to suffer the delay in retrieving them from disk drives. Because random-access memory would not hold an arbitrarily large number of pages, various algorithms were developed to determine which pages should be kept in random-access memory and when it was efficient to allow them to be replaced with other pages. All of this was done to operate the servers at peak performance.

44. For example, in 1996 it was feasible to equip a web server with 4 billion bytes (4 gigabytes) of random-access memory. This was at a time when most web pages included very

little in the way of memory-consuming graphics because many users were still accessing the web over slow dialup telephone connections. If a typical web page consisted of 10,000 bytes (a high estimate), then it would be possible to hold several hundred thousand such pages in main memory for very rapid delivery.

45. Another problem arose from the practical fact that every computer, no matter how fast, can only process a certain number of requests per second. If more are anticipated, a single machine will not be sufficient to provide adequate response. It was therefore necessary to employ more than one web server at a web site to handle the expected volume of traffic. Eventually, large collections of web servers, called "server farms," were deployed. The existence of more than one server led to the problem of determining which server a specific request should be set to, and how to get it there. One possible protocol was to keep track of the "load" on each server in the farm, that is, monitoring how much spare response capacity was available on each server at short intervals. A newly-arrived request would then be sent, or "dispatched," to the server that was least busy. This was one known method of "load-balancing," or trying to keep all the servers operating at approximately the same load.

46. Load balancing is an old art that developed long before the World Wide Web. It was initially used in multiprocessor systems (systems incorporating more than one computer) to determine what processing ought to be done on each machine to make effective use of multiprocessor power. Even load balancing itself, though old, developed from the much older theory of queuing, which dates back to the 1940s. Queuing theory attempts to estimate the optimal number of service points (such as checkout counters in a grocery store) that are needed to service anticipated but randomly distributed workloads. The same problem arises in multiprocessor systems and server farms, so the same technology is applicable.

47. Soon after the World Wide Web was developed, it became apparent that static pages were not adequate to satisfy users' information needs. A banking customer, for example, wants to see his own account data, not some general page of bank information. It is infeasible to store static pages containing all possible pages all bank customers might want to see and keep them

constantly updated. The solution was the development of the "dynamic page," a page that is not yet in existence at the time it is requested, but is created to order in real-time as needed. Dynamic pages were known long before the Patents were applied for.

48. A problem with dynamic pages is that they take time to create. The request must be analyzed to determine what kind of page is needed. Generally, data has to be retrieved from a database to populate fields on the page, which must then be formatted properly before it can be transmitted to the client. This process is vastly slower than delivering a static page out of random-access memory. In fact, it can be thousands of times slower.

49. The need to deliver dynamic pages caused huge degradation in the performance of web servers. The solution, known long before the Patents, was to analyze a request on the web server only long enough to determine whether it asks for a static or a dynamic page. If a static page is requested, the request can be fulfilled by the web server itself, by transmitting the pre-stored page, either from disk or from random-access memory. If a dynamic page is requested, the request is passed on to a different computer (a "page server") whose job is to retrieve data and construct the page. In most prior art systems, soon as the dynamic web page request is sent to the page server, the web server typically did not wait to be released by the page server to process other requests. It does not have to wait and do nothing in the interim. Indeed, the whole purpose of sending the dynamic page request to a page server is to avoid wasting the web server's valuable time and processing power.

50. When the page server completes construction of the page, it passes it back to the web server, which then delivers it to the client. The number of page servers required depends on the percentage of requests which are expected to be dynamic, and the time required to generate a dynamic page. If a very small fraction of pages are dynamically generated it might be sufficient to maintain only a single page server to support a large server farm. If almost all requests are for dynamic pages, then many more page servers will be required. The load-balancing technology used with page servers is no different from that used in web server farms.

51. At the time the application for the Patents was filed, April 23, 1996, the World Wide Web was already in public use for electronic commerce transactions and delivery of dynamic content. The Internet boom had begun, and Amazon.com had already been incorporated for two years. Web servers, server farms, load balancing and page servers were already well known and the problem purported to be solved by applicants had already been solved using the very methods claimed in the Patents.

52. The remainder of this section deals with various terms and concepts used in the Patents, with attention to their origins in the prior art. Certain terms discussed are used in claims that have not been asserted against Oracle in this action. The reason they are present is that because of the close relationship between the claims of the two Patents, I have used anticipation of unasserted claims by a reference to demonstrate anticipation of some Asserted Claims. For example, any reference that anticipates '335 claim 6 must anticipate '554 claim 5. That is the reason I have included a discussion of unasserted claims (and related terms) in this report.

a. The "multi-tier architecture"

53. On the Internet, the server load phenomenon caused great difficulty for popular websites. As a site's popularity (number of "visits," or browser requests) increased, the response time for each user also increased, so as sites became better known their performance suffered. In addition to the server farm, a different solution emerged to reduce the load on web servers. That was the idea of off-loading computational work, such as creation of dynamic web pages, from the web server itself and dispatching it to a different server, called the application server. The application server could then perform the work without impacting the performance of the web server. When the application server was finished generating a dynamic page, it would pass the page back to the web server, which would send it to the requesting client.

54. This architecture, known as the three-tier architecture, permitted web servers to confine their attention to the actual delivery of pages without spending costly computational time generating pages. In a three-tier architecture, the first tier is the client computer; the second is the web server; the third is the application server.

55. The problem of processing SQL database queries over the Internet was well-studied at the time of the Patents and led to the four-tier architecture discussed above in connection with Nguyen et al. (Reference 414). It often occurs in a three-tier architecture that tier 3 (the application server) needs to access a database in order to retrieve data. Because databases are normally closely controlled and secure against alteration, it is not feasible for each application server at tier three to maintain its own copy of the database. Therefore, a fourth tier, the database tier, was added to the architecture. An application server (tier 3) that requires information from a database must make a request to the database server (tier 4) to obtain it.

56. While the Patents introduce non-standard terminology for components of the four-tier architecture, such as "interceptor," "dispatcher," and "page server," the differences are linguistic only and the subject matter of the Patents is a straightforward use of the well-known multi-tier architecture. The inventions claimed in the Patents are not novel, regardless of the language in which it they may be expressed.

b. Dynamic web page generation

57. Systems for generating customized documents through database access predate the World Wide Web. Reference 059, Risberg et al. U.S. Patent 5,339,392, teaches a "software program providing a facility for a user to compose a custom active document using tools provided by the program. The active document can be comprised of one or more sheets each of which is composed in a custom manner by the user and each of which can show real time data and the changes therein generated by any complex system." (Abstract). The data described is stock quote information obtained over a network to provide real-time information. The Internet was in existence at the time, and qualifies as a "network" within the meaning of the reference. It would therefore be obvious to use the teachings of Risberg et al. to create web documents dynamically.

58. Dynamic web page generation, with data inserted into dynamic documents being retrieved from external databases, is the main focus of reference 153, "Oracle Web Server User's Guide (Release 1.0)." It "shows how the Oracle Web Listener handles requests for static and

dynamic documents, and how it returns documents to the client" and "how to use special utilities that generate HTML tags within PL/SQL procedures." (p. ii). The documents include "data stored in an Oracle database" (p. 1-2).

59. Dynamic web page generation is a principal object of at least the following references: 00J, 00Q, 039, 058, 141, 153, 161, 162, 172, 173, 180, 194, 195 and 196 (the "Dynamic Web Page References"), as shown in Exhibit 3.

c. load balancing

60. Even before the era of the Internet, it was recognized in client/server computing that the performance of servers degrades depending on the load placed on them. That is, if numerous clients make service requests at the same time, or if requests take excessive time to process, then the number of clients that can be served effectively can diminish severely. The Patents claim to address this problem: "Dispatcher 402 may determine that a number or all Page servers 404(1)-(n) are logged into Data source 408. In this scenario, Dispatcher 402 can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This 'load balancing' capability can significantly increase performance at a busy Web site." '554 6:14-18.

61. Load balancing itself was known in the 1970s.

62. After the development of the public Internet, extensive work was done on load balancing for web servers. Therefore, the relevant load balancing art is vast. In fact, 162 patents have been issued since 1976 on applications filed prior to the application date of the Patents dealing with managing load on web servers[4]. In addition, numerous load balancing papers appeared in scientific journals.

63. An early load balancing reference is Chow, et al., "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," which appeared in IEEE Transactions on Computers in May 1979. It applied queuing theory references from the 1960 to the dynamic load balancing problem and developed detailed solutions. In particular, when a job

[4] Obtained from the on-line search service of the U.S. Patent Office using the query "server" AND "web" AND "load" AND APD/1/1/1976->4/22/1996.

(request) is received at a dispatcher, it is sent based on dynamic information to a node (processor) that can more efficiently process the request.

64. In 1986, Rafael Alonso wrote an 88-page Ph.D. thesis at the University of California at Berkeley entitled, "Query Optimization in Distributed Databases Through Load Balancing." It explains load balancing methods in far greater detail than do the Patents, as do the following references: Hsu et al., "Dynamic Load Balancing in Homogeneous Distributed Systems," 6th Int'l. Conf. on Distributed Computing Systems (May 19-23, 1986); Liu et al. U.S. Patent 5,031,089, "Dynamic Resource Allocation Scheme for Distributed Hetergeneous Computer Systems," filed December 30, 1988; Bonomi et al., "Adaptive Load Balancing in a Nonhomogeneous Multiserver System with a Central Job Scheduler," IEEE Transaction on Computers, 39:10 (Oct. 1990); Lüling, et al., "Load balancing in large networks: A comparative study," Third IEEE Symposium on Parallel and Distributed Processing, Dallas, 1991; Lin et al., "A Dynamic Load-Balancing Policy with a Central Job Dispatcher (LBC)," IEEE Transactions on Software Engineering, 18:2 (Feb. 1992), pp. 148-158; Phillips et al., "Online load balancing and network flow," Proceedings of the 25th Annual ACM Symposium on Theory of computing, p.402-411, (May 16-18, 1993); Willebeek-LeMair, "Strategies for Dynamic Load Balancing on Highly Parallel Computers," IEEE Transactions on Distributed Systems, 4:9 (Sept. 1993); Eilert et al. U.S. Patent 5,537,542, "Apparatus and method for managing a server workload according to client performance goals in a client/server data processing system," filed April 4, 1994 (Reference 405); Civanlar et al. U.S. Patent 5,581,552, "Multimedia server," having an effective filing date of May 23, 1995 (Reference 403); Choquier et al. U.S. Patent 5,774,668, "System for On-Line Service in Which Gateway Computer Uses Service Map Which Includes Loading Condition of Servers Broadcasted by Application Servers for Load Balancing" (Reference 310), filed June 7, 1995; Mosedale et al., "Administering Very High Volume Internet Services," Ninth System Administration Conference, Monterey, CA (Sept. 18-22, 1995) (Reference 606); Schemers III, "lbnamed: A Load Balancing Name Server in Perl," Ninth Annual System Administration Conference (Monterey, CA, Sept. 18-22, 1995); Zaki et al., "Customized

Dynamic Load Balancing for a Network of Workstations," Univ. of Rochester Computer Science Department Technical Report 602 (Dec. 1995); Wallis U.S. Patent 5,740,371, "Load Balancing of Connections to Parallel Servers," filed March 18, 1996.

65. An important load balancing reference is 013, Garland et al., "Implementing Distributed Server Groups for the World Wide Web," Carnegie Mellon University Technical Report CMUCS-95-114 (Jan. 1995). Like many other prior art references from 1995 and earlier, Garland identified the problem of web servers handling too much load on a single server machine:

> "As the exponential growth of the World Wide Web continues with no near-term end in sight, the load placed on popular Web servers is often too great for a single server machine to handle. When the volume of requests to a server becomes prohibitively high the server tends to drop requests and eventually crashes. The service provider needs more computing power to meet the demands of the clients requesting the service. One solution is to simply buy a bigger machine that can handle the large load of service requests. This solution is, however, unacceptable; it provides only temporary relief from server overload. If server usage continues to grow rapidly, even the more powerful server will eventually be overwhelmed. This problem is likely to become more significant over the next few years as the number of commercial service providers and the size of the data objects being exchanged are likely to increase dramatically." See Introduction.

Garland also identified that the "obvious" solution to the above problem was to implement a distributed Web server architecture and dynamic load balancing techniques to manage incoming HTTP (page) requests by distributing them among server machines using a dispatcher server:

> "A fairly obvious solution to this problem is to implement a distributed Web server which will spread incoming request load among several machines. There is, however, a significant problem with this approach. The HyperText Transfer Protocol (HTTP/1.0) used by Web clients and servers identifies a resource on the Web using a valid hostname. Consequently, a distributed server needs to present a single hostname to maintain compatibility with existing Web clients. We can alleviate this problem by extending the concept of distributed process groups to the Web server level. . . . By supporting distributed Web server groups, we can provide a mechanism for smoothly scaling the server capacity of Web service providers. The use of server groups can reduce the latency of servicing a request provided there is sufficient available network bandwidth to accommodate higher request throughput. Our Web server group implementation can effectively balance request load, and is transparent to Web clients which support the HTTP/1.0 protocol." See Introduction.

Garland further describes a particular distributed web server architecture where a dispatcher server receives web page requests and dispatches them to appropriate member servers such that the dispatcher server and the several member servers can concurrently process requests:

> "We have chosen the following general architecture for our Web server groups: • There is a special server (the dispatcher) which coordinates and controls the server group. Requests intended for the server group are received by the dispatcher. • Attached to the dispatcher is a collection of member servers. These servers are the actual repository of data and fulfill requests made of the group. The dispatcher is the single entity representing the group to the outside world. Incoming requests are received by the dispatcher, and they are redirected to one of the member servers for actual processing." See Section 3.1.

66. The reference is agnostic as to the type of processing done by the member servers, which can include dynamic page generation. Garland further discloses the use of dynamic load balancing to dispatch the web page requests from the dispatcher server to the member servers according to dynamic load information exchanged between the dispatcher server and the member servers. See Figure 5 and Section 3.4 titled "Load Balancing." Because Garland also teaches the use HTTP to transmit web page requests, there would exist, according to epicRealm, TCP releasing between the member servers and the dispatcher server.

67. It is clear just from this brief list that the art of server technology embraced load balancing even before the World Wide Web, and the importance of load balancing to the Internet was recognized early.

68. Load balancing is specifically taught in connection with delivery of dynamic web pages in numerous references. These include 00W, 013, 016, 141 and the Oracle 1.0 References and the Oracle 2.0 References (together, the "Load Balancing References"), as shown in Exhibit 3. Because they teach load balancing and dynamic web page generation, it would be obvious to combine any of them with any system for dynamic web page generation that used more than one page server.

69. "Oracle 1.0" as used herein refers to an actual product used and distributed by Oracle Corporation. "Oracle 1.0 References" as used herein refers to a collection of documents which describe Oracle 1.0, including, without limitation, references 153 and 503/504.

70. "Oracle 2.0" as used herein refers to an actual product used and distributed by Oracle Corporation. "Oracle 2.0 References" as used herein refers to a collection of documents which describe Oracle 2.0, including, without limitation, references 604, 605 and "Oracle WebServer User's Guide Release 2.0 Production, Part No. A23646-1."

d. "releasing"

71. The precise meaning of "releasing" is a threshold issue in this case. Every Asserted Claim requires the page server to expressly release the web server or HTTP-compliant device.

72. Applicants chose to claim an express act of releasing. The Texas Court has interpreted "releasing," as used in the phrase, "said page server receiving said request and releasing said Web server to process other requests," to mean "said page server receiving said request and said page server performing an act (separate from merely receiving the request) to free the Web server to process other requests." It appears from various epicRealm papers that it believes that any of the following acts can constitute releasing: (1) an express release indication, e.g. a message or other communication overtly informing the Web server that it is being released; (2) a status indication that the request has been received; or (3) an indication that the page server has accepted the request and has agreed to process it. I believe that only (1) constitutes an act of releasing, but for invalidity purposes I have adopted epicRealm's theories.

73. The first form of releasing, an express release indication, is taught by L. Stein in the book *How to Set Up and Maintain a World Wide Web Site*, published in 1995 (Bates H-COM-P-ART 916-933). Stein's page server is a background process which generates a response to a request received by an HTTP server (Web server). Stein teaches a method of ensuring that the Web server does not remain tied up while the background process satisfies the original request. In particular, Stein discloses that "[t]he main tricky point when launching a background process from a CGI script is that the child process must close standard input, standard output and standard error in order to truly go into the background. Otherwise its connection with the HTTP server will remain open. This is why the first thing the child process does after forking is to close the filehandles STDOUT, STDERR and STDIN." (H-COM-P-ART 000924). Stein is

applicable as a reference in configurations where a web server is blocked because it has temporarily yielded control of one or more filehandles to a page server and is thus unable to use the corresponding files. The web serve in such a configuration would remain unable to process other requests until released by having the page server free the filehandles.

74. If the request is transmitted using TCP/IP, as is common when the Web server and page server are on different networked computers, the sender is provided with positive acknowledgement that the message has been received. Therefore, any transmission using TCP/IP meets the required limitation in the form (2) a status indication that the request has been received. Since HTTP relies on TCP/IP, the same acknowledgement is received if the message is transmitted using HTTP. Sending an acknowledgement it is not "an act (separate from merely receiving the request) to free the Web server to process other requests," as required by the Texas Court, for the simple reason is that the Web server is free to process other requests on its own and does not need to be "freed" to do so. The proof of this is that the Web server will continue to process other requests even if no acknowledgement is ever received.

75. Nevertheless, epicRealm has contended that the required step of "releasing" is satisfied by the TCP acknowledgement (this contention is hereinafter called "TCP Releasing"). The apparent justification for this assertion is that the result of the acknowledgment is that certain resources at both ends of the connection are no longer needed. Even if this is true, the use of those resources was not preventing the Web server from processing other requests, so releasing them does not have the effect of freeing the Web server to process other requests.

76. Even if epicRealm's TCP releasing theory is correct, the supposed "releasing" property of TCP is very old. The protocol is now substantially unchanged from its original version, published in Cerf et al., *Specification of Internet Transmission Control Program* in December 1974 as Internet Request for Comment (RFC) 675. ORCL01818208-8278. This publication describes TCP acknowledgement and even discusses a form of releasing on p. 29: "From the standpoint of controlling buffer congestion, it appears better to TREAT INCOMING PACKETS WITH HIGHER PRIORITY THAN OUTGOING PACKETS. That is, packet buffers

which can be released by copying their contents into user buffers clearly help to reduce congestion." Therefore, TCP acknowledgement and some form of releasing was included in the protocol in its original design.

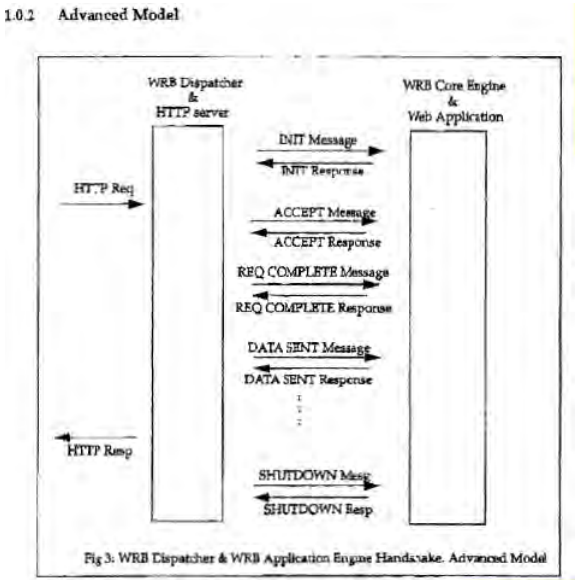
77. TCP acknowledgement has been discussed in many subsequent references, including Karn et al, *Improving Round-Trip Time Estimates in Reliable Transport Protocols***,** Proceedings of ACM SIGCOMM '87 (August 11-13, 1987) and Jacobson et al., *Congestion Avoidance and Control*, Proceedings of ACM SIGCOMM '88 (August 1988).

78. Any system that uses HTTP to pass requests to a page server performs TCP releasing, according to epicRealm's contentions, because HTTP relies on TCP as a data transport mechanism. The Common Object Request Broker Architecture (CORBA) provides an acknowledgement that the request has been received, which is analogous to TCP releasing and will be called "CORBA Releasing."

79. In this action, epicRealm proposes that "releasing" simply means "freeing." However, because "releasing" is recited as a positive claim step, this cannot mean implicit freeing, but requires an express act of freeing. That which was not tied up, however, cannot be "freed," so even express acts performed by the page server or messages communicated to the Web server cannot "free" it if was already processing other requests.

80. In the event that the Court construes "releasing" to mean implicit releasing, then every prior art reference cited herein performs such releasing.

81. Some prior art systems provide (3) an indication that the page server has accepted the request. This is true of Oracle's Web Request Broker (WRB), as described in U.S. Patent application 08/794,269. A drawing from a Section 131 affidavit filed in connection with that application, establishing a date of invention prior to March 5, 1996, is shown below.



82. Here there is a sequence of messages sent by the page server (Web Application) indicating that the web server has accepted the request. These include ACCEPT and REQ COMPLETE. In such a case it is not at all clear how these messages would accomplish any releasing, rather than simply acting as status messages. In the usual implementation, the web server would already be processing other requests when it receives these messages. However, this "message" releasing is equivalent to TCP Releasing according to epicRealm's contention because it is an acknowledgment of receipt.

83. If "releasing" means, as construed by the Texas Court, that the page server must perform "some act (separate from merely receiving the request) to free the Web server to process other requests," then inquiry is required into the nature of what that act might comprise. Presumably some indication would be passed back to the web server, either directly or indirectly, such as an acknowledgement of receipt of the request by the page server. For this to "free" the web server, however, the web server would have to be waiting for the indication before proceeding, which effectively requires an express act of releasing. For purposes of this report, I have performed an invalidity analysis under two separate theories of "releasing," both the TCP acknowledgement theory and express releasing.

84. The prior art teaches all three forms of "releasing": (1) express releasing as construed by the Texas Court and the other two forms of releasing advanced in epicRealm's contention: (2) acknowledgement of receipt of a request; and (3) indication that the page server has accepted the request. No matter which form of releasing is chosen, it was well known in the prior art, as was implicit releasing.

85. Recently, in the Texas Actions, epicRealm has advanced a new theory of releasing in the Texas Actions. It now says, in "Plaintiff epicRealm Licensing LP's Response to Defendants' Motion for Summary Judgment of Non-Infringement for Failure to Satisfy the 'Releasing' Limitation," (p.4) that "socket()-bind()-listen() calls" can constitute releasing. These "socket()-bind()-listen() calls" are calls made to the Unix operating system (only on Unix-based systems) that establish communications between two processes that will be exchanging data, even over a network. Socket() creates an endpoint for communications, bind() assigns this endpoint an address, and listen() prepares an addressed socket for incoming communications. Such a sequence of calls can be used, for example, to set up communications between a web server and a page server. Contrary to epicRealm's assertion, however, these calls do not release anything, and surely do not release anything in response to a request to generate a dynamic page. Sockets were added to Unix by the University of California at Berkeley in 1983, so were well known in the prior art. Although epicRealm has advanced this new theory in the Texas Actions, I'm not aware that epicRealm has advanced such a theory in this action against Oracle.

86. As described in the "Declaration of Peter Alexander, Ph.D.," filed in connection with the Texas Defendants' summary judgment motion in the Texas Action, the socket-related calls tie up resources instead of freeing (releasing) them. When a socket is established, a socket buffer must be allocated to hold data received by the socket (Alexander Declaration, p. 18).

87. In any event, the web server is not "released" in any sense by the execution of such calls because it was not being prevented form processing other requests in the meantime. It does not wait for any instruction to proceed – it simply proceeds. Since it is not being restrained, it cannot be "released." However, consistent with epicRealm's new contention, I have assumed

that "socket()-bind()-listen() calls" can constitute releasing, which I will refer to in shorthand as "SBL releasing."

88. None of the documents relied upon to show dates of conception or reduction to practice earlier than the effective filing date of April 23, 1996 mentions any act of releasing at all. These documents are at EPIC000001-173.

89. Andrew Levine, one of the inventors, testified at his deposition that when the Patents were filed, the page server performed no act of releasing: "Q -- did that product, did the page server of that product ever perform some act to release the web server after the page server received the request? A -- The page server never affected the web server itself. Q -- Okay. A -- The page server was a totally separate process. Its job was to sit there and receive requests, generate pages and return results. Period of the it was not an active thing to release the web server. Its I suppose an act ofo middle of anything occurring on the web server side, if that's what you're asking. Q -- So in the product that you were working on, there was never any type of message sent by the page server to anything that would act to release the web server so that the web server can continue processing? A -- The web server continued processing any ways. The web servers were all multithreaded web server who would sit there and continue to function to the best of their ability." Levine Depo. 83:22-84:20.

90. I note that the Patents recite sending a request for a web server to a dispatcher, then having the dispatcher forward the request to a page server, but then the page server releases the web server. This is difficult to understand since the page server and the web server are not in contact with one another, but are separated by the dispatcher. The Patents only make sense if the dispatcher is considered part of the web server.

91. epicRealm contends that several different types of "releasing" are found in the accused systems. I do not agree that any of them is releasing as construed by the Texas Court for the simple reason that they do not free the Web server because the Web server was already free and processing other requests prior to the releasing act. However, for purposes of invalidity analysis I will conditionally adopt epicRealm's various releasing contentions and show that

every claim is anticipated under one or more of epicRealm's varieties of "releasing." Every one of the references in Exhibit 3 utilizes implicit releasing in addition to any releasing it may perform under epicRealm's contentions.

e. "intercepting"

92. The Texas Court has construed "intercepting said request at said Web server" to mean "intercepting the handling of a request at a Web server."[5] It is my understanding that epicRealm has proposed the same construction for 'intercepting' in this action. This construction clarifies that intercepting involves not only redirecting the request itself, but also the processing (handling) of the request. My analysis below is also consistent with Oracle's proposed construction for the term "intercepting," namely, "receiving a request at the Web server and diverting the request before the Web server executable can process the request."

93. Intercepting is important to maintain server throughput. If a web server must spend any significant time analyzing a request before determining that it cannot handle the request, its efficiency will be diminished. It is therefore desirable to intercept such requests, ideally before they are even processed by the server.

94. Intercepting is inherent in any system in which the Web server processes some but not all, requests. Some mechanism must be used to discriminate between the requests the Web server will handle and which it will not. That mechanism which prevents the Web server from handling certain requests "intercepts" their handling within the Texas Court's construction.

95. Intercepting the handling of requests was well known in the prior art, as discussed in connection with Eckerson, above. Intercepting CGI requests (used to generate dynamic web pages) is taught in reference 166, Bittinger et al. U.S. Patent 5,754,774: "This web server originated communication is intercepted by the server-side intercept module 40 and transformed by a client/server specific data stream." (10:12-14). "After receiving the information from the client-side intercept module 30, the server-side intercept module 40 checks its server cache resident in the second computer to determine if a server cache entry exists corresponding to the

[5] There is a parallel construction for "intercepting said request at said HTTP-compliant device."

URL of the HTTP request originated by the web browser." (12:25-29) Bittinger et al. contains 421 references to intercepting.

96. Intercepting requests to a web server for dynamic page generation prior to their handling by the web server was well known in the prior art. Reference 164, Chiu et al. U.S. Patent 5,752,022, "Method for creating a hypertext language for a distributed computer network," literally teaches intercepting such a request: "The present invention includes a server-based application which intercepts an initial request for access to documents which may contain imbedded references to linked information." (3:11-14).

97. There is extensive teaching of intercepting in reference 153, "Oracle Web Server User's Guide (Release 1.0)," which discloses a "Web Listener" that performs the function of intercepting: "When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created "on the fly" by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." (p. 1-3).

98. At least the following references teach intercepting in the context of dynamic web page generation: 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 039, 051, 058, 066, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 602, 603, 604/605 and 607 (the "Intercepting References"), as shown in Exhibit 3.

f. "routing"

99. epicRealm has proposed in the Texas Actions that "routing" means "sending or forwarding data along a path towards a destination." In this sense routing is a very old concept form the earliest days of packet-switched networks, of which the Internet is a late example. Within such networks, the conventional method of causing information to be sent from one machine to another was to route it over the network. Routing is therefore completely conventional and obvious as a means of network data transmission. My analysis below is also consistent with Oracle's proposed construction for the term "routing," namely, "transmitting."

100. Nevertheless, more specific kinds of routing were known in the prior art. Reference 154, "Discover: A Resource Discovery System based on Content Routing," deals entirely with the movement of web documents (content) by routing. It states, "[a] Content Routing System consists of a hierarchical network (a directed acyclic graph) with conventional information providers at the leaves and content routers as mediating nodes. … The end-point information providers support query based access to their documents. At a content router, a user may browse and query the information about providers registered at that content router as well as make use of the router's facilities for query refinement and routing." (p. 4).

101. Therefore, routing documents, and specifically content routing, was known in the prior art.

g. "dispatching"

102. epicRealm has represented in the Texas Actions that "dispatching" means "examining a request to make an informed selection of which page server should process the request based on dynamic information maintained about page servers, the dynamic information indicating which page server can more efficiently process the request, and sending the request to the selected page server." That is, it means more than simply transmitting, but requires an informed decision involving examination of the request. The decision can be based on server load, which is conventional load balancing, or may be based on the data sources available to the various page servers, or both. A page server that cannot obtain the necessary data cannot process a request at all, let alone process it efficiently. My analysis below is also consistent with Oracle's proposed construction for the term "dispatching," namely, "analyzing a request to make an informed selection of which page server should process the request, and sending the request to that page server."

103. It is an unresolved question whether a system having only a single page server can perform dispatching. If there is only one page server, then what meaning can be ascribed to an "informed selection"? With only a single page server, there cannot be any "selection" at all, let alone an informed one. Furthermore, information about server load is irrelevant when only a

single page server is available. It is my opinion that systems having only a single page server cannot satisfy the limitation of "dispatching."

104. My opinion on this point is bolstered by the fact that in the case of a single page server there is no need to examine the request to determine which page server should process the request. The request must be examined to learn whether a dynamic page is being sought, but that examination is not performed a part of dispatching, but intercepting, which is recited as a separate step. My opinion is further bolstered by the fact that in a configuration having only a single page server, there is no "dynamic information maintained about page servers" which can be used to inform the choice of page server, since there is never more than one option.

105. Nevertheless, epicRealm has accused certain systems having only a single page server of infringing claims containing dispatching limitations. Therefore, I will assume for purposes of invalidity analysis that prior art references teaching only a single page server also teach dispatching, which I will refer to as "Single-Server Dispatching."

106. Under epicRealm's proposed construction, dispatching requires consideration of dynamic information maintained about page servers. Dynamic information is information that can change and is not an inherent feature of the architecture. Information does not need to change on a minute-to-minute basis to be dynamic. Some systems maintain files, sometimes referred to as configuration files, which maintain information about page servers such as their network addresses and the resources available to them. These configuration files can be changed at any time, and the test for use of dynamic information is not the frequency with which the information changes, but whether a change to the configuration file affects the choice of page server made by the dispatcher. Because epicRealm has accused of infringement certain systems which make use of configuration files for dispatching, for purposes of invalidity analysis I will adopt epicRealm's contention and will refer to such dispatching herein as "Configuration File Dispatching."

107. Dispatcher processes are taught in reference 104, "Web Server Technology" at pp. 151-153. A dispatcher process is one that selects a "helper process" to perform a task. The

dispatcher must maintain dynamic information about the helper processes to avoid attempting to hand off a task to one that is busy. The architecture described was available as a commercial Netscape product in 1994 (p. 153).

108. Dispatching is taught in reference 153, "Oracle Web Server User's Guide (Release 1.0)," in connection with generating dynamic web pages. The reference teaches a Web Agent that performs the function of a dispatcher, which must decide which database server to connect to in order to retrieve the information needed to generate the web page. "In order to connect to an Oracle7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (p. 5-4)

109. Dispatching in this manner is also taught in references 013 and 370. However, any reference which makes a specific decision which server to pass a request to, including various forms of load balancing, also teaches dispatching.

110. Reference 153 makes the express statement that SQL*Net may optionally be used between the Web Agent and the Oracle7 Servers (page 1-3). SQL*Net is described in the reference "Understanding SQL*Net, Release 2.2," Part No. A32090-1 (Oracle Corp. 1995) ORCL01806620-6818. SQL*Net fundamentally allows the creation of distributed networked databases that reside on disparate database servers (page servers), but that appear to the user as a single database: "SQL*Net allows Oracle tools and applications to access, manipulate, share and store data in Oracle databases residing on remote servers. In addition, SQL*Net enables data access between multiple database servers." ORCL01806623. Of course in a configuration having multiple page servers there must be a mechanism to decide which one should receive a request. Beginning at page ORCL01806663, the SQL*NET reference describes how dispatching occurs via a network listener. When the network listener receives a request it has the option to either (1) spawn a dedicated server and route the request to it, (2) route the request to a shared dispatcher of a multi-threaded server, or (3) route the request to one of the prespawned dedicated servers depending on the nature of the request and dynamic information maintained about the various servers. Figure 2-3 at ORCL01806664 illustrates this process:

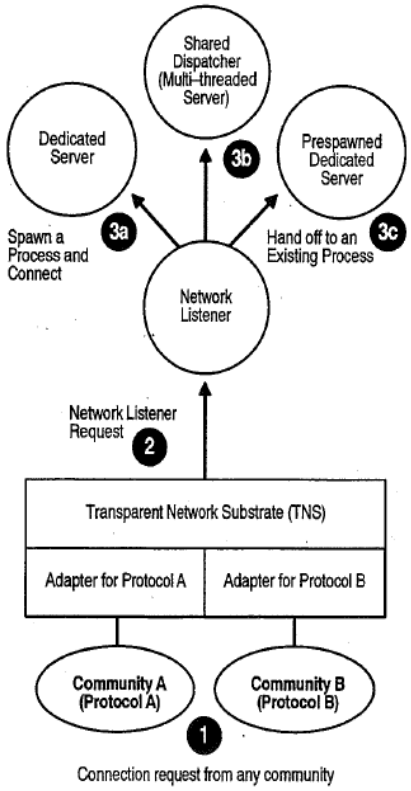


Figure 2 – 3 Network Listener in SQL*Net Connection

Beginning at page ORCL01806665, the SQL*NET reference describes how SQL*NET establishes connections and dispatches requests to particular prespawned dedicated servers. Additionally, at pages ORCL01806667-669, the SQL*Net reference describes how a request to a "multi-threaded server" is handled using shared dispatchers: "The listener receives the connection request, performs the connection handshake and determines if the client is allowed to connect … The listener issues a redirect message back to the client containing the address of the least-called dispatcher that is listening on the protocol used by the client … The listener and dispatcher perform a short handshake to update each other of the presence of a new connection. This is so that the listener can load balance connections between dispatchers." ORCL01806668. The request must be examined to determine which protocol it requires. The dynamic information maintained about page servers is their level of utilization (load). An informed

selection is made based on this dynamic information, so SQL*Net performs dispatching as construed by the Texas Court.

111. Release 2.3 of SQL*Net offered a more advanced form of load balancing. Release 2.3 of SQL*Net was available at least as early as February 1996, as demonstrated in "Oracle7 Server Migration, Release 7.3, Part No. A32540–1 (Oracle Corp., February 1996), page 9-28: "Pre-7.3 dispatchers cannot contact multiple listeners with the new initialization parameter, MTS_LISTENER_ADDRESS. Therefore, if you wish to use load balancing, you must upgrade to SQL*NET, Release 2.3 and Oracle7, Release 7.3." A technical description of Oracle's load balancing can be found in "Understanding SQL*Net, Release 2.3," Part No. A42484-1 (Oracle Corp. 1996), ORCL01808597-8820: "Listener Load Balancing. The listener load balancing feature provides for the distribution of connections among listeners for Oracle7 Servers. By having connections distributed among a number of listeners, no single listener is likely to be overburdened, and connection time will be faster. Once randomly selected, the listeners behave as they have in previous releases, distributing connection requests to the dispatchers on the basis of their load (for multi-threaded servers) or to dedicated servers. How Listener Load Balancing Works. There can be multiple listeners for a single database or multiple listeners for two or more equivalent databases. Multiple Listeners for a Single Database. You can have more than one listener for a single database of any kind. The listeners for a dedicated server must be on the same node as the server. Listeners for a multi-threaded server can be distributed on different nodes; the dispatchers are able to register with listeners across nodes." p. 2-16. Further, "All dispatchers register with all the listeners listed in the database parameter file, and the client randomizes between the listeners. The listener passes the address of the least-used dispatcher to the client, and the client connects using that dispatcher." p. 2-17. This method of load balancing implements "dispatching" as construed by the Texas Court.

112. Dispatching is not possible without "intercepting," for if the request is not rerouted before the Web server processes it, the request will never be dispatched. Therefore, the references which teach dispatching in the context of dynamic page generation also teach

intercepting. Dispatching is taught in at least the following references: 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 019, 051, 058, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607 (together, the "Dispatching References"), as shown in Exhibit 3. It is also taught separately in "Understanding SQL*Net, Release 2.2" and "Understanding SQL*Net, Release 2.3," two of the Oracle 1.0 References.

h. "data dynamically retrieved from one or more data sources"

113. Every Asserted Claim claims "data dynamically retrieved from one or more data sources." Of course, the very notion of a dynamic web page implies that data must be retrieved dynamically to form the page; otherwise, a static web page would suffice. In any event, retrieving data dynamically for inclusion in a web page was well known in the art of dynamic page generation and is described in numerous references.

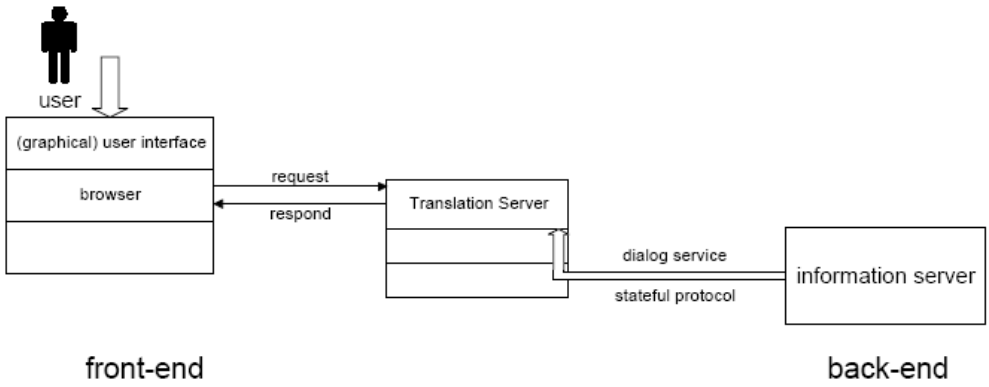
114. Reference 058, "Providing Customers Information Using the WEB and CORBA", teaches merging data obtained from databases into a dynamic web page.

115. Reference 062, "Interface to Relational Databases," teaches a system for generating dynamic web pages that include the results of queries processed by database systems running on remote servers. "Each time a request is received, the server inspects it to determine if it is a standard HTTP 'GET' request for a simple document or a Pythia-specific HTTP 'QUERY' request. GET requests are handled in a conventional manner - the requested document is sent back to the client. QUERY requests, however, are actually SQL database queries wrapped up with additional 'template' information. The template describes how to take the results of the query and format it graphically or textually for the user. A QUERY request still produces a valid HTML document as its result - but the document is completely built 'on the fly' from the results of the database query." (p. 1).

116. Reference 063, Clausnitzer, "A WWW Interface to the OMNIS/Myriad Literature Retrieval Engine," teaches retrieval of data from a remote database for inclusion into a dynamically construed web page: "Each OMNIS-HTTP server is in permanent connection with

its OMNIS database and translates incomming [sic] requests into database queries. The resulting data is then transformed [sic] back into HTML and returned to the WWW client." (p. 8)

117. Reference 067, "Translation Servers: Gateways Between Stateless and Stateful Information Systems," teaches dynamic page generation with data obtained from remote databases. Fig. 5 from the reference is shown below.



118. "The Translation Server must be able to process HTTP-requests from the front-end and to transform them into a dialog-type communication with the back-end. Upon a request from a front-end, it parses the schema and composes responses in hypertext markup language ([HTML], as used by W3) to get additional information from the user (Fig. 7, line 3). Then it accesses the stateful information server to get the requested information. The result is translated on-the-fly by the information server into HTML and returned like any other W3-object, most presumably as a HTML-document." (p. 4)

119. Reference 102, Poole et al. U.S. Patent 6,006,242, "Apparatus and method for dynamically creating a document," teaches assembly of a dynamic web page by retrieving data elements from a database.

120. Reference 104, "Web Server Technology," teaches assembling a dynamic web document from information stored in pieces in databases: "the Web gateway program must create a single Web document in an acceptable format before sending it to the browser." (p. 252).

Fig. 6.14 shows a web (httpd) server invoking a CGI script that interrogates a database to assemble a custom document.

121. Reference 164, Chiu et al. U.S. Patent 5,752,022, teaches that dynamic data retrieval was part of the prior art even at the time Chiu et al. was filed: "FIG. 1 illustrates a block diagram of existing WORLD-WIDE WEB data linking. A user 101 inputs a request 102 for a particular document or other piece of information . The request is sent through normal WORLD-WIDE WEB 103 interfaces (e.g. WebExplorer, etc.) to search, locate and retrieve the requested document. Once the document, as shown in exploded view 105, is retrieved the user may select highlighted text 106 which may be of further interest. The highlighted text, in HTML format, includes hidden imbedded links 107 to other related information 108/109 as prepared by hypertext authoring tools. Upon selecting a highlighted text section, the system will retrieve the related data through the Web and return it to the user in much the same manner as the process for obtaining the original request." (2:20-34). This is an elementary form of dynamic data retrieval.

122. Dynamic data retrieval and insertion into dynamic web pages is taught in at least the following references: 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 039, 051, 058, 066, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607 (together, the "Dynamic Retrieval References"), as shown in Exhibit 3.

i. "connection cache"

123. The purpose of a connection cache, according to the Patents, is to reduce the time needed to establish connections to data sources: "One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. 4, Page server 404(1) can retrieve data from data source 406, data source 408 or data source 410. Page server 404(1) can maintain connection cache 412(1), containing connections to each of data source 406, data source 408 and data source 410, thus eliminating connect times from the Page servers to those data sources." '554 6:56:65. This stated purpose is consistent with Oracle's proposed construction for the term "connection cache," namely, "a store

of information identifying open connections to data sources for eliminating subsequent connect times to those data sources."

124. However, this form of connection caching was not new. It is taught, for example, in reference 051, "Harvest: A Scalable, Customizable Discovery and Access System." Harvest is a system for accessing information over the Web and assembling it into an HTML document (dynamic web page): "The Query Manager uses an HTML form to collect the query from the user, and then passes the query to the Harvest gateway. The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker's database and to the original resource described by the summary object" (p. 15). Connection caching is taught expressly in the reference: "As noted earlier, Harvest allows information to be collected remotely so that it can interoperate with sites that do not run the Harvest software. Because this poses so much more load on Provider-site servers, we implemented a connection caching mechanism, which attempts to keep connections open across individual object retrievals" (p. 10). The "objects" from which retrieval is being made are data sources, and the connection cache keeps a set of open connections to these data sources.

125. Connection caching is taught in reference 00Q (discussed below in connection with '554 claim 4) and reference 348 (discussed below in connection with the obviousness of all claims). It was well known in the art and routinely used in software that connected to multiple sources simultaneously.

126. Reference 016 teaches on page 109 that "Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections," which is maintenance of a connection cache to data sources.

127. Reference 141 teaches maintaining multiple Hyper-G session, which have connection to data sources and having the ability to "distinguish requests from different WWW

clients and to assign them to already opened Hyper-G sessions" (Section 1 of the reference). The list of "already opened Hyper-G sessions" is a connection cache.

128. Reference 501 teaches the use of Microsoft's Open Database Connectivity (ODBC), which maintain a connection cache, in conjunction with dynamic page generation.

129. SQL*Net teaches connection caching. Multiple connections to Oracle7 Servers (which access data sources) are maintained, and the system is aware at any time which of these connections are live at any given time: "This message shows that since this listener has been running it has made total of five connections using prespawned server for TCP/IP and has one running at this time. It also shows that there are seven connections currently established using multithreaded server dispatcher with total of 30 established since the process started." ORCL01806717.

130. Connection caching is taught in at least the following references: 002, 00O, 00Q, 00W, 016, 019, 051, 141, 153, 161, 162, 172, 180, 195, 501/502 and 604/605 (together, the "Connection Cache References") and separately in "Understanding SQL*Net, Release 2.2," one of the Oracle 1.0 References, as shown in Exhibit 3.

j. "logging in"

131. "Logging in" to a data source is claimed in '554 claim 6 and '355 claims 6 and 20. Logging in to a data source has two meanings in the art. Oracle has proposed that "logging in" means "presenting credentials that allow access to," such as a user name and password. The specification of the Patents provides no explanation of "logging in," and epicRealm has not proposed any construction for the term "logging in" in this action. If, however, "logging in" means simply "connecting" to a database, then logging would occur anytime any system accesses a database.

132. "Logging in" in the strict sense is taught in reference 153, "Oracle Web Server User's Guide (Release 1.0)": "A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service … The Oracle Web Agent is a CGI program that the Web Listener executes when a request is

received for a dynamic document. This program logs into the database and executes stored PL/SQL procedures that have been specified as part of the URL." (p. 1-4). "Thus, the Human Resources department and the Accounts Payable department use different username/password combinations to log on to the database. If both want to write CGI applications using the Oracle Web Agent to access their data, the Web Agent needs to use the correct username/password when it logs on to the database." (p. 5-5).

133. Both meanings of "logging in" were known to those of skill in the art, and the need to establish a connection and possibly present authentication credentials were part and parcel of access to data sources. It is a distinction without a difference to argue whether logging into a database system is the same as logging into a "data source" itself. Data sources are accessed by database systems and are typically maintained as files or database tables. On a technical level one does not "log in" to a file, either in the Patents or in the prior art. One logs into a system which then allows access to data sources. Therefore, for purposes of invalidity analysis, I have treated logging into a database system which manages a data source as logging into a data source. Alternatively, one may regard the database system as a data source itself.

134. SQL*Net, used with Oracle WebServer 1.0, teaches logging into data sources: "The connect operation is initiated during any standard database login between the client application and the server with information such as the client machine name and user name being passed to the remote machine." ORCL01806661.

135. The SQL*Net reference actually teaches at least three additional ways of logging into data sources. One is through logins preauthorized by operating system settings: "In SQL*Net version 1, Oracle provided support for operating system authorized logins across a network using the special database login." ORCL01806736. A second way is through "Secure External Authentication Logins": "Using the network identity provided by the authentication service the Authentication Logins database can provide secure external authentication over a nonsecure protocol such as TCP/IP." ORCL01806738. A third method is through proxy login: "Proxy authentication (secure database links, that is, secure server-to-server connections) will be

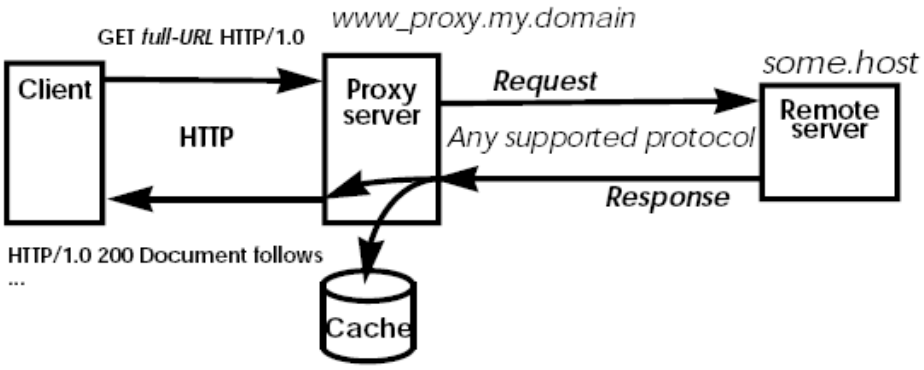
used when user who has been externally authenticated by network service attempts to use an anonymous database link one without username or password credential that can make use of a proxy login. When connection is requested, the credentials are passed from the local database server to the remote server." ORCL01806738.

136. Logging into data sources (or the database systems that access them) is taught in at least references 002, 00K, 010, 016, 019, 141, 153, 172, 173, 180, 195, 196, 501/502, 604/605 and 607 (together, the "Logging In References"), as shown in Exhibit 3.

k. "page cache"

137. Caching of web pages for various purposes was well known in the prior art. Caching is an expedient to reduce network load and processing time. Browsers cached web pages so if the user requested them again it would not be necessary to have them retransmitted over the network. Web servers cached pages in main memory so it would not be necessary to retrieve them from hard disk or some other server, thus reducing latency time. Dynamic pages were cached to avoid the expense of regenerating them. All of this was amply taught in the prior art. My analysis of page caching in this report is consistent with Oracle's proposed construction for the term "page cache," namely, "a memory that stores finished Web pages which the page server uses to fulfill requests instead of re-accessing the one or more data sources."

138. Reference 066, "World Wide Web Proxies," from April 1994, contains extensive teachings on caching web pages. Fig. 5 from the reference is shown below. Section 2.2 of the reference deals with server-side caching.



139. Reference 104, "Web Server Technology," contains a 46-page chapter entitled, "Web Document Distribution and Caching." It explains the main justification for caching pages: "the Web server itself must serve all the requests, no matter how many times the same document is delivered to the same place." (p. 190). "Web servers themselves may act as caches." (p. 193) A family of devices known as "caching servers" are described beginning on p. 194. The use of proxy servers as document caches is taught on pp. 194-203. The fact that a web page may have changed since the last time it was requested (e.g. for dynamic pages) is discussed expressly in the reference.

140. Page caching is taught in reference 051, "Harvest: A Scalable, Customizable Discovery and Access System," which contains over 60 references to caching and includes an entire section entitled, "The Object Caching Subsystem" (pp. 18-19).

141. Page caching of dynamically generated web documents is taught in reference 056, Boutell et al, "Techniques for Server-Side Dynamic Document Generation": "Techniques for dynamic HTML and GIF document generation will be discussed, with particular emphasis on the REF52 2D gel database system and the gd dynamic GIF creation library. Strategies for fast user response without race conditions, such as the caching of documents in a virtual space and aggressive pre-caching of likely requests, will be presented." (p. 1) "By storing the complete response to previously requested locator URLs and throwing out old responses on a least-recently-used basis, it is possible to quickly evaluate whether the desired view has already been generated and stored in the cache, and deliver it to the user immediately without the need to calculate its actual contents." (p. 4).

142. Page caching of dynamic web pages is taught in Section 3.3.5 of reference 141.

143. Page caching of dynamic web pages is also taught in Reference 113, "The Case for Geographical Push-Caching," which mentions "how to cache dynamic pages, such as those generated by cgi-bin scripts that are used to create Web pages on the fly." (p. 54).

144. Reference 140, "Semantic Data Caching and Replacement," is devoted entirely to page caching.

145. Caching of web pages is taught in reference 153, “Oracle Web Server User's Guide (Release 1.0), which contains 38 references to caching: “There are three options for specifying files in the file cache • An explicit filename can be given. • An incomplete filename ending in an asterisk can be used to indicate that all files that match the portion before the asterisk will be cached. • A directory can be specified, in which case all its files and subdirectories will be cached.” (p. 3-14).

146. Reference 166, Bittinger et al. U.S. Patent 5,754,774 contains extensive teachings on web page caching: “After receiving the information from the client-side intercept module 30, the server-side intercept module 40 checks its server cache resident in the second computer to determine if a server cache entry exists corresponding to the URL of the HTTP request originated by the web browser. If, after interrogating the web browser originated communication as reflected in block 125, the server-side intercept module 40 determines that a cache entry does exist corresponding to the information requested by the web browser originated communication the "Yes" branch of block 125 is taken. The server-side intercept module 40 then compares the current date and time of the SSI module 40 to the sum of the SDT of the server cache entry corresponding to the information requested by the web browser originated communication and the predetermined client coherency time interval received from the client-side intercept module.” (12:25-41). This teaching reflects the prime object of server-side page caching, which is to avoid generating a page that has already been generated. Bittinger et al. contains 431 references to page caching.

147. Page caching of dynamically generated web pages is taught in at least references 002, 00H, 016, 051, 066, 141, 153, 161, 196, 501/502, 601 and 604/605 (together, the “Page Cache References”), as shown in Exhibit 3.

l. “HTML templates,” Custom HTML extension templates”

148. A “template” in the context of dynamic page generation is a pattern for a page that can be instantiated to form an actual page by filling in markers in the template with dynamically retrieved data. The specification of the Patents states that these markers can determine both the

physical location of the data on the page and the place at which the data can be found. It refers to "customized HTML templates that specify the source and placement of retrieved data." '554 6:36-37. "My analysis below is also consistent with Oracle's proposed construction for the term "custom HTML extension templates," namely, "custom files using non-HTML markers for specifying where data from data sources should be inserted to create an HTML file."

149. An "HTML template" is a template that includes HTML tags. When data is filled in between such tags, the browser rendering the document for display will format the data according to the tags. Thus the tags determine the format in which the data will be viewed.

150. Because HTML is tag-based, any HTML template is automatically a "tag-based text template."

151. HTML templates were old at the time the Patents were filed and were extensively taught in reference 150, Leaf et al. U.S. Patent 5,754,722. Claim 22 of Leaf refers to a "data processing system having a HyperText Transport Protocol (HTTP) server for responding to requests from external browser programs" and goes on to claim "(a) establishing a hyper-text markup language (HTML) template that is associated with a predetermined function" and "(h) creating an HTML document from the HTML template."

152. An "extension" to a computer language refers to the addition of one or more new constructs to the language that were not part of its specification. Many references teach HTML extensions, in which new tags are added that are not HTML. These extensions cannot be understood by standard browsers because browsers implement the official HTML specification. Therefore, some further processing must be done on a file which contains HTML extensions to remove them or convert them to standard HTML before the page is delivered.

153. An "HTML extension template" is a template which includes HTML extensions.

154. Leaf teaches: "The HTML template file is used in creating an HTML document that is returned to a Web Browser. The transaction gateway client reads data values from a data buffer returned from the transaction processing system (as specified by the mapping file), the data values are written to appropriate locations in the HTML document as directed by

information contained in the HTML template file." (3:17-24) A full HTML template is shown in Fig. 7. "FIG. 7 illustrates a portion of the HTML Template 314 created for the View Definition 316 of FIG. 8. The fbnames from the View Definition have place holders in the HTML Template for inserting data values, either by the Transaction Gateway Client 40 or by the user of the Web Browser." (8:60-67) This is "the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates," as claimed in '335 claim 9.

155. The reference states, "The writing of a data value to the proper location in the HTML document is accomplished with an HTML Template, e.g., 314." (11:52-55). Such templates are "tag-based text templates for configuring said Web page," as claimed in '335 claim 8. The HTML template is also a "tag-based text template" as claimed in '335 claims 11, 22 and 25.

156. Reference 00A, Filepp et al. U.S. Patent 5,347,632, teaches "page template objects," which are HTML extensions, in the context of dynamic page generation. Reference 00H, Dönszelmann, K. Rodden, Gateways for World-Wide Web (1994), teaches extending HTML by adding an additional tag which is replaced by dynamically retrieved data as a page is generated. Thus a page containing such a tag is an HTML extension template. Reference 00K, Sjölin, A WWW Front End to an OODBMS (1994), teaches a "general presentation descriptor" (GPD) which is able to generate HTML in dynamic page production and hence is an HTML extension template.

157. Reference 00Q on p. 5 teaches HTML files which are marked with "designated strings" which are replaced with dynamic data in generating a page, hence are HTML extension templates. Reference 00T teaches extending HTML to allow embedded database queries. The tags thus added are replaced by dynamic data when a page is generated. Reference 016 on p. 111 and Reference 062 contain similar teachings, of extending HTML to allow embedded SQL queries.

158. Reference 058 contains extensive teaching of "Meta HTML files," which are HTML extension templates used in dynamic page generation. Reference 063 teaches a "Mask

Definition Language," which is used for creating HTML extension templates. Reference 137 teaches an entire extension mechanism to HTML called Hyper-G. When a Hyper-G document is processed to insert dynamic data, a regular HTML document is produced for delivery to a client.

159. Reference 153 teaches on p. 3-2 that "HTTP is an application level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many systems, through extension of its request methods (commands)." And on p. 6-37: "The design of the hypertext procedure and function packages allows you to use customized extensions. Therefore, as the HTML standard changes, you can add new functionality similar to the hypertext procedure and function packages to reflect those changes." This teaches customized HTML extension templates.

160. Reference 172, a Microsoft document, on page 6 expressly teaches HTML extension templates in generating dynamic web pages, as does Reference 173, discussed below with respect to '335 claims 13 and 14.

161. Reference 181, Kremen et al. U.S. Patent 5,706,434 teaches a language called Extended HTML (XHTML) which allows creation of HTML extension templates in dynamic page generation. References 194 and 195 expressly teach a "server-side HTML extension known as the group extension" used in generating dynamic web pages from templates.

162. I note that there is no requirement that a template, HTML template or HTML extension template be a static document or file. An executable procedure (script or computer program) can constitute a template, and there is no teaching to the contrary in the Patents. As an example, a program while executing might access data dynamically and write text to its output stream that comprises the retrieved data surrounded by HTML tags. The fact that a program is used to apply a predetermined pattern to the data does not make it any less a template.

163. HTML extension templates in the context of dynamic web page generation are taught in at least the following references: 002, 00H, 00K, 00Q, 00W, 016, 019, 039, 058, 141,

153, 172, 173, 194, 195, 196, 401, 501/502, 601, 604/605 and 607 (together, the "HTML Extension References"), as shown in Exhibit 3.

m. "tag-based text templates"

164. Tag-based text templates are antique, dating back at least to SGML (Standard Generalized Markup Language), a standard on which HTML and XML were based.

165. Meta HTML text tags are taught in reference 058, "Providing Customers Information Using the WEB and CORBA." The reference deals with creation of web documents in real-time by a "Production Application Server" (page server) which obtains data from data sources for substitution into text tags. The form that contains these tags are templates used to form web pages. "The OWL remote method returns multiple lines, string arrays, of tag/value pairs that contain the results. The tags in the returned data correspond to the META HTML tag names. These are substituted according to the following rules to create the HTML output." (p. 6) The reference goes on to teach the use of variables in HTML for which specific values can be substituted to produce a dynamically generated document.

166. Reference 062, "Interface to Relational Databases," teaches a system for generating dynamic web pages that include the results of queries processed by database systems running on remote servers. The retrieved data is inserted into tag-based text templates to form the final delivered page. "QUERY requests, however, are actually SQL database queries wrapped up with additional 'template' information. The template describes how to take the results of the query and format it graphically or textually for the user. A QUERY request still produces a valid HTML document as its result - but the document is completely built 'on the fly' from the results of the database query." (p. 1)

167. Reference 102, Poole et al. U.S. Patent 6,006,242, "Apparatus and method for dynamically creating a document," teaches assembly of a dynamic web page by retrieving data elements from a database for substitution into tag-based text templates. "A 'tag' is commonly referred to as descriptive markup. A tag is a code that is inserted in a document to identify the information contained in a logical element of the document. Tags delimit (mark, surround, or

enclose) an element in an SGML file. An SGML document generally contains both tags and text." (3:62-66).

168. Since HTML templates are "tag-based text templates," all references teaching the former, such as reference 150, also teach the latter.

169. Tag-based text templates in the context of dynamic web page generation, are taught in at least the following references: 002, 00H, 00J, 00K, 00Q, 00W, 016, 018, 019, 039, 051, 058, 141, 153, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607 (together, the "Tag Template References"), as shown in Exhibit 3.

n. "drives a format of the data"

170. Reference 150, Leaf et al. U.S. Patent 5,754,722 teaches the use of HTML templates (tag-based text templates) to drive (determine) a format of data: "A predetermined HyperText Markup Language (HTML) template file is also established for each desired service. Each of the predetermined mapping files sets forth the format and content of the data buffer that is used for communicating between the transaction gateway client and the on-line transaction processing system for the particular service. A mapping file directs the transaction gateway client where to write data to and read data from the data buffer for predetermined fields of the database. The HTML template file is used in creating an HTML document that is returned to a Web Browser. The transaction gateway client reads data values from a data buffer returned from the transaction processing system (as specified by the mapping file), the data values are written to appropriate locations in the HTML document as directed by information contained in the HTML template file." (3:9-24). The HTML template thus determines the format of the data in the final generated HTML document (web page), as claimed in '335 claims 10 and 24. In note again that a template can be stored file, a script or a computer program.

171. Reference 153, "Oracle Web Server User's Guide (Release 1.0)" expressly teaches the use of HTML templates to determine a format of data: "Body tags are used in the main text of your HTML page. They can format a paragraph, allow you to add hidden comments to your text, and add images within the body of your HTML text." (p. 6-11).

172. Using templates to drive a format of data in a dynamically generated web page is taught in at least the following references: 002, 00H, 00J, 00K, 00Q, 00W, 016, 018, 019, 039, 051, 058, 141, 153, 162, 172, 173, 180, 194, 195, 401, 501/502, 601, 604/605 and 607 (together, the "Format Driving References"), as shown in Exhibit 3.

o. "object handling extension"

173. An "object handling extension" is claimed in '335 claims 13, 14, 27 and 28. The term is not used at all in the Specification. The term but appears to be a generic noun of which "OLE extension" (used in claims 14 and 28) is the only disclosed species. (The invalidity of claims 13 and 27 on this basis is discussed in the section "Non-Art Invalidity," below.) "Object handing extension" is not a term of art, is not explained in the Patents, and is therefore indefinite. In fact, a Google search reveals that the only occurrence of "object handling extension" on the World Wide Web is in the '335 Patent. Nevertheless, for the purpose of invalidity analysis, I have assumed that "object handling extension" means an extension to a system that allows it to handle objects. Thus any prior art reference in which objects are accessed to produce a dynamic web page involves processing an object handing extension.

p. "OLE extension"

174. OLE, which stands for "Object Linking and Embedding," is a Microsoft technology which allows applications to access each other's objects. An "OLE extension" is software that permits an application to invoke OLE.

175. OLE in the context of web document generation is taught on p. 4 of reference 051, "Harvest: A Scalable, Customizable Discovery and Access System" as follows: "The database community has done a great deal of work with more structured distributed information systems, but to date has fielded no widespread systems on the Internet. Similarly, a number of object-oriented systems have been developed (e.g. CORBA [25] and OLE [30] but have yet to be deployed on the Internet at large." The citation to [30] refers to Microsoft's OLE 2.01 Design Specification. The reference clearly teaches using OLE to gain access to object over networks.

176. OLE is taught with respect to the common gateway interface in reference 104, "Web Server Technology": "Web servers on Microsoft Windows platforms can use the standard DLL interface (Brain 1994) or OLE (Brockschmidt 1995)."

177. Peden et al. U. S. Patent 5,615,325, "Graphical viewer for heirarchical [sic] datasets," having a filing date of September 29, 1994, teaches a system for viewing web pages: "The spokes viewer may be utilized as an element in products such as personalized news casts, Internet information managers, corporate databases, etc." (4:65-5:2). "The spokes node specifies data specific information (file name, story, etc.), attributes (font, color, etc.), as well as location and relationship information. The spokes annotations 320 operates on objects that are not inherent in the base data. For example, the annotation data may be associated with a node, or may be attached as a general annotation. Examples of annotations are text, circles, rectangles, lines, audio, video, hypertext, hypertext mark up language (HTML) and object linking embedding (OLE) annotations." (7:39-42).

178. References 172 and 173 expressly teach OLE in the context of dynamic web page generation. Reference 172 states on p. 7, "To deliver the full potential of the Active Internet, Microsoft recently introduced ActiveX technologies, a robust framework for creating interactive content using software components, scripts, and existing applications that can be embedded in HTML documents. Specifically, ActiveX technologies enable developers to build Web content easily using ActiveX controls (formerly OLE Controls), ActiveX scripting, and ActiveX documents." OLE Controls are taught as a means of "creating interactive content," which includes dynamic web pages. Reference 173 describes dynamic Web page generation with HTML extension templates and teaches that Microsoft's browser supports OLE, which informs the reader that OLE can be relied upon in generating pages since the browser will be able to handle OLE extensions: "Microsoft's Internet client is The Internet Explorer; as of this writing 2.0 is out, and 3.0 in beta. Microsoft is committed to making this an extension of Windows, so that you might browse the Web as naturally as you currently might browse your hard disk. Regardless of features, inclusion of this product with the Windows OS will be a fairly

compelling argument for its use. Java will be supported in the Explorer, as will distributed OLE objects."

179. Reference 401, Anand et al. U.S. Patent 5,692,181 teaches use of OLE 2.0 for dynamically generated web documents: "Reports are compound documents that display data from a database in text and graphics (e.g., graphs, tables). Reports are the result of running a Smart Report Definition. Smart Reports are in the HTML format and are OLE 2.0 compliant." (4:40-44).

180. The use of OLE in generating dynamic web pages is taught in at least Peden et al. and References 172, 173 and 401 (the "OLE References"), as shown in Exhibit 3.

q. "dynamically updating"

181. The nature of database queries, particularly SQL queries, is that they can cause updating of the database being queried at the same time the query is being processed. Such dynamic updating is part and parcel of relational database handling. Nevertheless, various references dealing with dynamic page generation specifically teach updating of data sources.

182. Reference 150, Leaf et al. U.S. Patent 5,754,722, teaches that "[t]he Transaction Processing System processes the request against the Database 28 according to the specified request (e.g., select, update, delete). The Transaction Processing System returns data and/or status information to the Transaction Gateway Client, which in turn formats the data into an HTML document that is forwarded to the Web Server. The Web Server sends the HTML document to the requesting Web Browser." (4:60-67).

183. Reference 153, "Oracle Web Server User's Guide (Release 1.0)" expressly teaches dynamic updating in connection with dynamic web page generation: "The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 Server that enables the creation of *dynamic* HTML documents from data stored in an Oracle database. When the data changes, these HTML documents are updated automatically, with no further effort on the part of the site administrator." (p. 1-2). The reference specifically explains how the data in a data source can be updated as part of a query: "Which SQL statements determine the actual data content? Chiefly, three —

INSERT, UPDATE, and DELETE. INSERT places rows in a table, UPDATE changes the values they contain, and DELETE removes them." (p. 4-8). The reference goes beyond simple updating, allowing the user to set triggers to cause dynamic documents to be updated automatically if any underlying data on which they depend should change in the database.

184. Reference 157, Lagarde et al. U.S. Patent 5,721,908 teaches dynamically updating a data source that is used to provide information for a dynamic web page. It describes a "DIS server" which obtains data dynamically over the Web for inclusion in a hypertext document: "The DIS capsule is an object program with executable additions which we have created to interact with the control program. It should be also understood that the DIS capsule object can perform programmable functions on data which is retrieved from databases. Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved. It can act on the data to create new data and basically act as a dedicated processor processing data gathered or created during a Web browser request to output the end result to the user under programmable parameters determined by the creator of the DIS capsule, as they may be selected, if desired, by the user as pad of the request." (16:5-17).

185. Dynamic updating of data sources during data retrieval in generating dynamic web pages is taught in at least the following references: 016, 019, 039, 051, 153, 162, 163, 172, 173, 180, 194, 195, 401, 501/502, 601, 604/605 and 607 (the "Dynamic Updating References"). Furthermore, because updating is one of the core functions of Structured Query Language (SQL), any system that allows SQL queries to generate dynamic pages teaches dynamic updating.

r. "page server"

186. epicRealm has proposed that the term "page server" be construed to mean "page-generating software that generates a dynamic Web page," and that "Web page" be construed to mean "Web content displayable through a Web browser," which are the constructions arrived at by the Texas Court. Therefore, according to epicRealm, a page server is software that generates dynamic Web content displayable through a Web browser. One of skill in the art would

understand that a Web page has not been "generated" until it is complete; that is, if data is to be retrieved and surrounded by HTML tags so it can be sent to a browser, then the page has not been generated until the tags have been added.

187. epicRealm appears to take the position that any dynamic data whatsoever, even if it is not a complete web page suitable for delivery to the user and is not the page that was requested, is nonetheless a "Web page" for purposes of infringement simply because a Web browser would be capable of displaying it. epicRealm has accused an item of software known as MySQL as being a page server. MySQL is a database management system capable of responding to SQL queries for dynamic data. Normally the data produced by a MySQL query is inserted into an HTML template or otherwise surrounded by HTML tags to turn it into a completed Web page. Logically, the software which finishes making up the page is the page server. However, epicRealm seems to be arguing that any software that it used in the preparation of a page is a "page server" because its intermediate output could be (but is not) displayed on a Web browser. I do not agree with this position. However, for purposes of invalidity analysis, I will refer to a page server which produces unfinished dynamic pages as a "Partial Page Server" based on epicRealm's contentions.

188. I understand that epicRealm takes the position that, despite the teaching in the patent against multi-threading, software running on a single computer, including a web server, interceptor, dispatcher and page server, falls within the ambit of all claims. Oracle disputes this position. Indeed, contrary to epicRealm's proposed construction, Oracle's proposed construction for "page server" reflects the asserted patents' teaching that the page server must operate on a separate machine (i.e., use a separate processor) from that of the Web server in order to realize the claimed benefits of the claimed inventions. I will adopt epicRealm's position for purposes of invalidity analysis, although I disagree with it and do not concede its correctness. To the extent, however, the Delaware Court adopts Oracle's proposed construction, it was known and obvious to one of ordinary skill in the art to offload web page requests from a Web server to a page server running on a separate machine from the Web server such that the Web server and page server

could process requests at the same time. See, e.g., the discussion on multi-tier architectures as discussed in subsection "a" above and the discussion of load-balancing in subsection "c" above.

s. "HTTP-compliant device"

189. The term "HTTP-compliant" was known in the prior art to mean a device or program that comports with the Hypertext Transfer Protocol. Dozier et al. U.S. Patent 5,870,552, "Method and apparatus for publishing hypermedia documents over wide area networks," filed March 28, 1995, teaches a system for publishing web documents. It states, "Both NaviPress and NaviServer are thus 'open' in the sense that any http-compliant client computer can access most services provided by NaviServer, and NaviPress can be used with any http-compliant WAN server." (16:61-65) It was thus known to regard web servers as HTTP-compliant devices.

190. The term "HTTP-compliant device" is not and was not a recognized term of art. The only references to it that are contemporaneous with the Patents that I have been able to find are the Patents themselves. The fact that it is not a term of art does not mean that it is not understandable to one skilled in the art, however. It means a device that implements, i.e. "complies with" the HTTP protocol.

191. Chow et al. U.S. Patent 6,029,175, "Automatic retrieval of changed files by a network software agent," having an effective filing date of October 26, 1995, makes reference to "HTTP-compliant systems": "Specifically, the Revision Manager makes it possible to add services to all HTTP-compliant systems, including services that provide registration of interests, updating intervals, change monitoring, automatic notification of changes, and display of updated information." (38:55-59).

192. It was obvious to refer to a web server as an "HTTP-compliant device" and therefore to generalize any process involving dynamic web page generation and web servers to use HTTP-compliant devices.

193. I understand that epicRealm now contends in the Texas Action that an HTTP-compliant device can be something known as a "Layer-7 switch." Normally, network plumbing

like routers and switches operate simply by examining the destination addresses of messages. This is analogous to the processing of mail at the Post Office. The mail sorters do not read the letters – they make their sorting decisions based only on the address written on the envelopes. However, there is no hint or discussion of Layer 7 switches in the Patents. They did not become commercially available until 1999[6].

194. Under epicRealm's theory, the Layer 7 switch plays the role of a Web server because it receives the initial page generation request from a Web client. The Layer 7 switch then "dispatches" the request to one of a set of Web servers, which then plays the role of being a "page server" within the meaning of the claims. Under epicRealm's theory, the Layer 7 switch performs the act of "examining the request," as required for dispatching. Layer 7 switches can also perform load balancing. Even assuming that denominating the "HTTP-compliant device" to be a Layer 7 switch and the "Web server" to be a page server is technologically justified, nevertheless the resulting architecture is no different from the three-tier architecture discussed in connection with Eckerson, above.

ALL ASSERTED CLAIMS ARE ANTICIPATED

195. All of the Asserted Claims are anticipated. The number of references that establish anticipation is large, in some cases as many as 30, and the number of combinations that establish obviousness is in the thousands. To simplify the presentation, I have arranged the references in the form of a spreadsheet, attached hereto as Exhibit 3, which is an integral part of this report. The columns of the spreadsheet are labeled with claim limitations, adopting the numbering scheme from the Texas Actions. The rows of the spreadsheet list references, with three-character identifiers corresponding to the numbering of references in Exhibit 1[7]. Exhibit 3 contains almost 3000 cells mapping 99 claim elements to 30 references.

196. The anticipation results are summarized in Exhibit 4. The rows correspond to the Asserted Claims. The three-character codes in the columns identify anticipating references for

[6] Wilson, "Layer 7 Switch Takes Prioritization from the Top," InternetWeek (April 5, 1999).
[7] Effort has been made to conform the reference numbering to that of the Defendants in the Texas case and the Plaintiffs in the Delaware case, but small inconsistencies have made this impossible.

each claim using the numbering of Exhibit 1. The last column gives the number of established anticipating references for each row.

197. Exhibit 5 is a summary of which claims are anticipated by various references, including individual claim elements[8]. The first set of references, labeled from 002 to 198, correspond to the labeled references the Texas Defendants' Contentions. The second set, labeled starting at 300, correspond to the unnumbered references in the Texas Defendants' Contentions. The third set, labeled from 400 to 503 onward, are references from the Texas Actions not listed in the Texas Defendants' Contentions. References 501-503 are from the Reexamination Requests. References 601-607 are from the present action.

198. It is possible to read off the anticipating references directly from Exhibit 4. All in all, I count 409 pairs of claims and anticipating references.

199. Claim element numbering in this report refers to the numbering scheme in Exhibit 1. While effort has been made to conform the numbering to various submissions and claim charts of the parties, it has not been possible to do so in each instance because inconsistent numberings have been used.

200. In the spreadsheet of Exhibit 3, text in a cell means that the claim limitation in that cell's column is taught in the reference of the cell's row. A green cell means that I have not found the claim element corresponding to the cell's column in the reference corresponding to that cell's row. The citations to prior art in the cells are exemplary only. No attempt has been made to list all places a particular element is taught in each reference. The entries in Exhibit 3 are to be read as a supplement to the material in the body of this Report.

201. In Exhibit 5, I have indicated by sequences of red cells bearing the letter "X" claims that are anticipated by various references. An "X" in a cell that is not red indicates that the corresponding element is taught by the reference, but that I have not found all elements of the same claim in the reference. The absence of an "X" does not necessarily indicate that the corresponding element is missing from the reference, merely that I have not established an

[8] Certain other references cited in this report are not numbered and do not appear in Exhibits 3, 4 or 5 because I have not asserted that they anticipate any claims of the Patents.

explicit correspondence. In many cases this is simply a consequence of the huge number of element/reference pairs. The total number claim elements is 99 (43 in '554 and 56 in '335). In practice, over 200 references were considered[9]. The product of 99 and 200 is 19,800, a completely unmanageable number of element/reference pairs. Even considering only the 16 Asserted Claims, these still comprise 43+33 = 76 elements.

202. Of the total of 99 claim elements, only 30 are substantially distinct. (Of the 76 elements of the Asserted Claims, 23 are distinct.) These distinct elements are designed by white cells in row 15 of Exhibit 3. Cells shaded tan represent claim elements that are substantially identical to other claim elements.

203. A claim is anticipated if all of its limitations and those of its parent claims are found in a single reference. Anticipated claims are indicated by solid horizontal red bars spanning one or more cells of the spreadsheet of Exhibit 5.

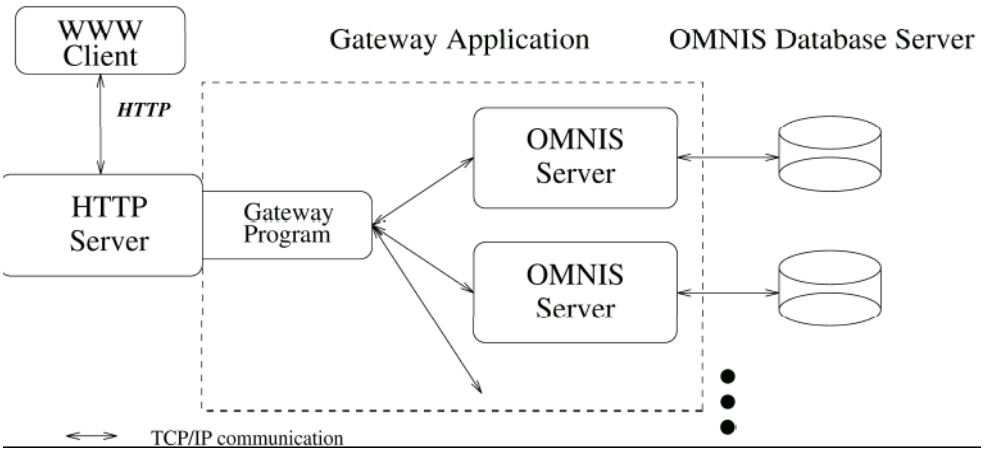
204. To illustrate the number and diversity of anticipating references, I will show that each of the 16 Asserted Claims[10] is anticipated by using several different references as possible.

'554 Claim 1

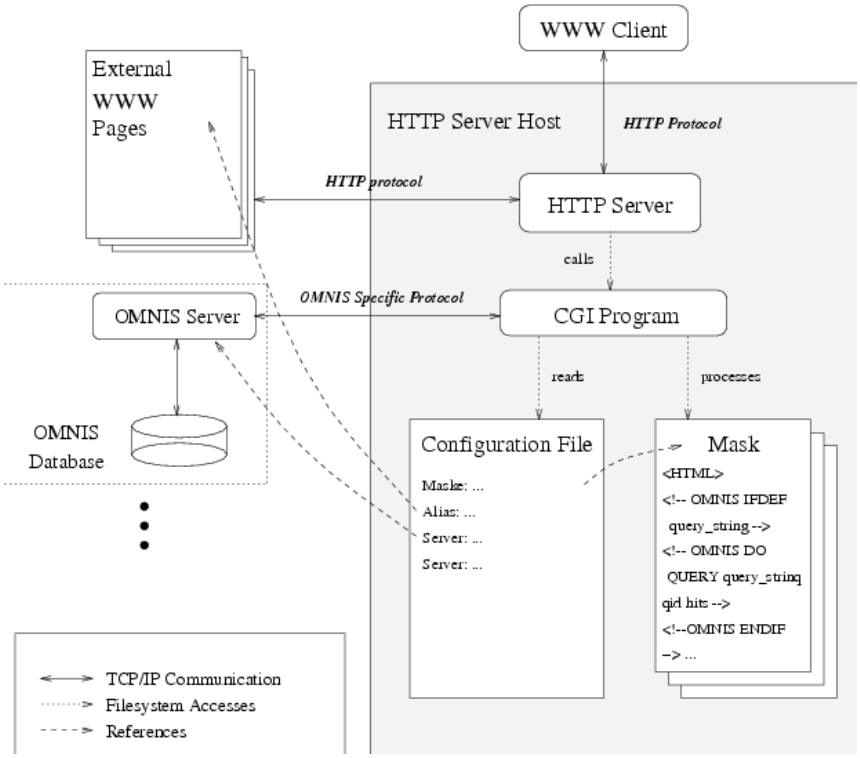
205. Claim 1 is anticipated by reference 063, Clausnitzer, "A WWW Interface to the OMNIS/Myriad Literature Retrieval Engine," which teaches generating dynamic web pages from databases: "Each OMNIS-HTTP server is in permanent connection with its OMNIS database and translates incomming [sic] requests into database queries. The resulting data is then trasformed [sic] back into HTML and returned to the WWW client." Sec. 3.2. Thus the preamble 1a is met.

206. Figure 5 from the reference is shown below. It shows a four-tier architecture. Tier 1 is the WWW Client (browser). Tier 2 is the HTTP (Web) server. Tier 3 includes the OMNIS Servers, which function as page servers. Tier 4 includes the OMNIS Database Servers, which provide dynamic data to the OMNIS servers.

[9] Many more references were examined, but mapping to claim elements was considered in only about 200 of them.
[10] The number of claims actually asserted is 14. However, claims 2 and 16 of the '335, Patent depend from claims 1 and 15, respectively, so the number is actually 16 for purposes of invalidity analysis.



207. Further details of the architecture are given in Fig. 6 from the reference:



208. Limitation 1b1 is met because the web server (labeled in the figure as HTTPD – Server) routes dynamic requests to the Gateway Program, which in turn sends them to the "OMNIS Server" (page server). The page server is actually a combination of the OMNIS Server

with that portion of the Gateway Program which, according to the reference, transforms it [the page] into a WWW client readable format." (p. 8).

209. The OMNIS Database can also be considered a page server under the Partial Page Server Contention.

210. Limitation 1b2 is met because the OMNIS Server (page server) receives the request or it could not process it. (In general limitation 1b2 is trivially met in every reference for the same reason.)

211. Limitation 1b3 is met because the page server (OMNIS Server) communicates with the web server via TCP/IP, as shown in the figure. This provides acknowledgement to release the web server, according to epicRealm's contention re TCP releasing.

212. Limitation 1b4 is met because the handling of the request is intercepted at the HTTP (Web) Server and the request is instead sent to the Gateway Program.

213. Limitation 1b5 is met because a portion of the Gateway Program serves as a dispatcher and dispatches the request to the appropriate OMNIS Server.

214. Limitation 1b6 is met because dispatching occurs as construed by the Texas Court. Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to the correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2. Thus the reference teaches both Configuration File Dispatching and dispatching by load balancing.

215. Limitation 1c is met because the HTTP Server is free to processes other requests as the OMNIS Servers are creating pages. The load which would normally fall on the HTTP Server is sent instead to the least-loaded OMNIS Server.

216. Limitations 1d1 and 1d2 are met because a web page is dynamically generated in response to the request, as shown in Fig. 5 of the reference, and the generated page includes data dynamically retrieved from at least one data source, identified as "OMNIS Database Server."

217. Because reference 063 incorporates all the limitations of claim 1, it anticipates at least claim 1.

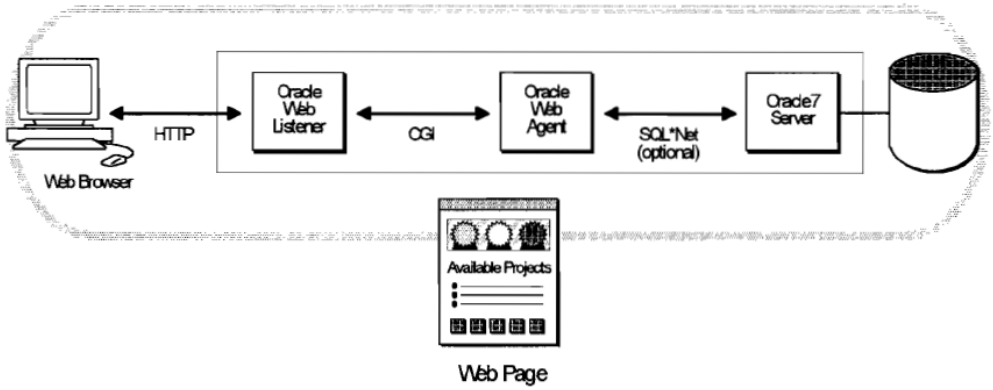
218. As discussed below, '554 claim 1 and '335 claim 2 are virtually identical. Therefore, '554 claim 1 is anticipated by any reference that anticipates '335 claim 2, namely at least references 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 051, 058, 063, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607.

219. Further, claim 1 is anticipated by references 153 and 503/504, the Oracle 1.0 References and the Oracle 1.0 product. Reference 153 is entitled "Oracle WebServer User's Guide Release 1.0." It can also be found on pages MAC003037-3284 of Appendix D to the '554 Reexamination Request. Reference 503 is "The Oracle WebServer: A Technical Discussion," which can be found as Exhibit E to the '554 Reexamination Request. These documents can be considered a single publication because both emanate from the manufacturer, Oracle Corporation and both describe the same product. Further, reference 503 specifically mentions Release 1.0 of Oracle WebServer, which is the subject of reference 153. Reference 503 does not bear Bates numbers so page references to it herein refer to actual page numbers in the document.

220. Reference 153 describes a database system some features of which are claimed in reference 169, Bookman et al. U.S. Patent 5,761,673 (the "Bookman Patent"). The Bookman Patent was considered by the examiner during the prosecution of the Patents. However, Reference 153 contains matter that did not appear in the Bookman Patent, and is not cumulative to Bookman. The U.S. Patent and Trademark Office on May 4, 2007, in ordering a reexamination of the '554 Patent "The Oracle references raises [sic] a substantial new question of patentability as to claims 1-3 and 5-11, which question has not been decided in a previous examination of the 5,895,554 patent to Lowery." The "Oracle references" referred to in the order includes reference 153.

221. Reference 153 is 248 pages long. The entire Bookman patent, including drawings, is only 9 pages long. There are numerous disclosures in reference 153 that are not present in Bookman. Bookman, for example, makes no mention of SQL*Net, which, as seen below, is expressly disclosed in reference 153 as being an optional component. SQL*Net utilizes load-balancing dispatching, which is not taught in Bookman. Reference 153 teaches TCP releasing, according to epicRealm's contention, which in not taught in Bookman. While concurrency is inherent in Bookman (simultaneous processing by the web server and page server) it is expressly taught in Reference 153. Reference 153 expressly teaches multiple data sources, while Bookman teaches one database. SQL*Net, available with Oracle 1.0 as described in Reference 153, used connection caching, which is not taught in Bookman. Reference 153 teaches page caching; Bookman does not. Bookman teaches updating a data source as part of a request inherently, through SQL. Reference 153 teaches updating expressly (MAC003112). Thus the teachings of Bookman and Reference 153 are considerably different.

222. The Oracle WebServer structure is also discussed in more detail below in connection with '335 claim 15. A diagram of its overall architecture, from page MAC003053, is reproduced below.



223. Briefly, the web server in Oracle is called the web listener, which receives requests. Requests for dynamic pages are intercepted and dispatched by a web agent to Oracle7 Server (page server), which accesses the necessary SQL databases to retrieve dynamic information used to configure the web page.

224. Reference 153 states that "The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database" (MAC003052). Thus the Oracle 1.0 References teach the preamble 1a.

225. Reference 153 states that "The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." (MAC003052). Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram. Thus the Oracle 1.0 References teach step 1b1.

226. The diagram shows that the request is received by the Oracle7 Server (page server). Thus the Oracle 1.0 References teach step 1b2.

227. Reference 153 states that "For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." In this context, "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. If the Web Listener and the Web Agent are running on the same physical computer, then when the Web Agent hands the request off to the Oracle7 (page) server using HTTP, TCP releasing is performed, according to epicRealm's contention, to free resources on the computer on which the Web Agent is running. Since the Web Agent is on the same computer as the Web Listener (server), under epicRealm's theory these resources are also freed for the Web server. Thus the Oracle 1.0 References teach step 1b3.

228. The web listener (web server) intercepts dynamic page generation requests without processing them. Reference 153 states that "When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is

created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Requests for static pages are handled by the web server itself. Requests for dynamic documents are intercepted (not processed by the web server) but handed off. Thus the Oracle 1.0 References teach step 1b4.

229. The Web Agent in OracleServer acts as a dispatcher. Reference 153 states that "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054. "In order to connect to an Oracle7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use. This information is stored in the owa.cfg file (located in the OWS Administrative directory) as part of a Web Agent service. The Web Agent determines which service to use by parsing the SCRIPT_NAME environment variable, which is set up by the Web Listener in accordance with the CGI 1.1 specification." MAC003146. Thus the Oracle 1.0 References teach the limitations of steps 1b5 and 1b6 by configuration file dispatching. Oracle 1.0 also optionally uses SQL*Net, as shown in the figure. Therefore it teaches SQL*Net dispatching.

230. The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates "asynchronously." "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Thus the Oracle 1.0 References teaches steps 1c and 1d1.

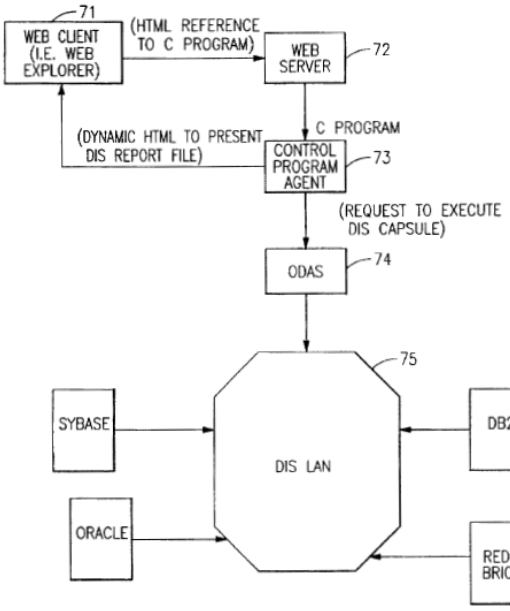
231. The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. Thus the Oracle

1.0 References teach step 1d2. Because they teach all the steps of claim 1, they anticipate claim 1.

232. Claim 1 is anticipated by at least 29 references, as listed in Exhibit 4.

'554 Claim 2

233. Claim 2 is anticipated by reference 180, Rogers et al. U.S. Patent 5,701,451. To show anticipation of claim 2 it is also necessary to show anticipation of '554 parent claim 1. Fig. 7 from the reference is shown below.



234. The reference teaches a system for fulfilling dynamic web browser requests. Claim 16 states that the process dynamically creates new HTML statements for a report of results according to a request of the web browser." Therefore, the preamble 1a is met.

235. The dynamic page request is passed to a control program agent 73 (Fig. 7), which in turn routes (sends) it to a "data interpretation system (DIS) server" 75 (through an ODAS 74), which obtains the necessary dynamic data to create the page. The page is passed back to the control program agent, which adds the necessary HTML elements and sends the page to the web server 72 for delivery to the web browser 71. The "page server" function of generating the

dynamic web page is performed by a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Limitation 1b1 is met because the request is routed (sent) from the web server 72 to the control program agent 73 (page server).

236. ODAS, the DIS LAN and each data source connected to DIS LAN 75 each by themselves are Partial Page Servers, hence are also page servers according to epicRealm.

237. Limitation 1b2 is met because the page server cannot process the request unless it receives it. The receiving is shown in Fig. 1 of the reference (MAC003667).

238. Limitation 1b3 is met because TCP/IP is shown as the mechanism for sending the request to the DIS in Fig. 11 and is described at 19:13-19. Hence the reference teaches TCP releasing.

239. Limitation 1b4 is met because the handling of the request is intercepted away from the web server and not processed by the web server, but instead is sent to the control program agent 73, as shown in the drawing.

240. Limitations 1b5 and 1b6 are met because the request is routed (sent) to the ODAS 74, which dispatches it to the DIS (page server). This is Single-Server Dispatching, according to epicRealm. If the individual data sources connected to DIS LAN 75 are regarded as page servers, then ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the 'hidden' defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having

an application processing agent which has the information desired." 12:49-61. The "generalized search" calls for dynamic information.

241. Limitation 1c is met because the web server 72 processes other requests while the page generation request has been passed off to the DIS. The DIS is a separate machine on its own LAN and therefore concurrent processing by the Web server is inherent in the disclosed architecture.

242. Limitation 1d1 is met because a dynamic web page is generated. 4:52-55.

243. Limitation 1d2 is met because the page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client … ." 5:45-51.

244. The limitation of claim 2 is met because the DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.

245. Because reference 180 incorporates all the limitations of claims 1 and 2, it anticipates at least those claims.

246. Further, claim 2 is anticipated by the Oracle 1.0 References. The Oracle system must determine which database to access to complete the dynamic page. Reference 153 states that "After connecting to the database the Web Agent invokes the appropriate PL/SQL procedure, whose name is obtained by parsing the PATH_INFO environment variable." (MAC003146) Thus the data sources to be used are identified by examining the PATH_INFO environment variable, which meets the limitation of claim 2.

247. Claim 2 is anticipated by at least 29 references, as listed in Exhibit 4.

'554 Claim 3

248. Claim 3 of the '554 patent is subsumed by claim 4 of the '335 patent. Claim 4 of '335 includes all the limitations of '554 claim 3. Therefore, any reference that anticipates '335

claim 4 anticipates ’554 claim 3. It is shown in Exhibit 3 that ’335 claim 4 is anticipated by references 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 051, 058, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607. Therefore, ’554 claim 3 is also anticipated by those references. Nevertheless, it will be shown expressly that ’335 claim 3 is anticipated by reference 141, Derler “The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services.” published in March 1995. To show anticipation of claim 3 it is also necessary to show anticipation of ’554 parent claims 1 and 2.

249. The reference teaches a computer-implemented method for managing a dynamic Web page generation request to a Web server, meeting preamble 1a. “When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items.” Sec. 3.3.4.

250. The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3. This satisfies the limitation 1b1 of routing a request from the web server to the page server.

251. Receiving the request by the page server is taught expressly in the reference: “This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session.” Sec. 4.2.1. This meets the limitation 1b2.

252. Releasing occurs because Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a

client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to epicRealm's contentions. Thus limitation 1b3 is met.

253. The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to deal with it. Sec. 4.2.2. This meets limitation 1b4.

254. The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5. This meets limitation 1b5.

255. The selection of page server made by the dispatcher is based on dynamic information maintained about page servers: "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2. Load balancing to make a selection of an efficient page server requires dynamic information about page servers, so limitation 1b6 is met.

256. Concurrency is disclosed because the HyperG server processes the request while the web server processes other requests at the same time: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the 'outside' is not burdened

with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2. Thus limitation 1c is met.

257. The Hyper-G system dynamically generates a Web page in response to the request. See Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a dynamic page is shown in Fig. 4.13. Thus limitation 1d1 is met.

258. The data sources are the "databases" of the reference. "Unlike the WWW, every document stored in a Hyper-G database can be searched for. As attributes (title, keywords, creation date, etc.) are stored in a separate database, searching for these attributes can be performed efficiently." Sec. 3.5. "The gateway uses HTML fill-out forms to provide a user interface for searches in Hyper-G databases." Sec. 4.5.5. The results of the searches, containing data dynamically retrieved from the database, are furnished to the user in HTML. This meets limitation 1d2. The reference therefore anticipates claim 1.

259. Reference 141 also anticipates claim 2. Identification of data sources is expressly taught: "The Hyper-G server, in processing the request, identifies one or more data sources from which to retrieve data. Sec. 3.3.4. "The Hyper-G server is able to store 'remote objects' (i.e. Hyper-G objects in the link database that are pointers to remote systems) for accessing WWW, Gopher and WAIS servers. This allows Hyper-G clients to contact such remote systems and presenting a 'Hyper-G view' (as far as possible) of information on non-Hyper-G servers." Sec. 3.3.4. Thus the limitation of claim 2 is met and claim 2 is anticipated.

260. Reference 141 anticipates claim 3. During the process where the Hyper-G system dynamically generates a web page, the Hyper-G server dynamically retrieves data from one or more data sources, i.e., the Hyper-G server's database or other data source. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"); Figure 4.13. Also Section 3.3.4. This meets the limitation of claim 3.

261. Because reference 141 incorporates all the limitations of claims 1, 2 and 3, it anticipates at least those claims.

262. Further, claim 3 is anticipated by the Oracle 1.0 References, which were previously shown to anticipate '554 claims 1 and 2. Reference 153 states that "The Oracle WebServer is an HTTP server with a tightly integrated Oracle 7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." (MAC003052). It further states, "The PL/SQL procedure executes, generating an HTML document. With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." Thus the Oracles References teach the step of claim 3.

263. Claim 3 is anticipated by at least 29 references, as listed in Exhibit 4.

'554 Claim 4

264. Claim 4 of the '554 patent is subsumed by claim 5 of the '335 patent. Claim 5 of '335 includes all the limitations of '554 claim 4. Therefore, any reference that anticipates '335 claim 5 anticipates '554 claim 4. In particular, '554 claim 4 is anticipated by reference 00Q, Antchev et al., "A WWW Learning Environment for Mathematics," which was published in December 1995 in conjunction with the Fourth Annual WWW Conference held in Boston. To show anticipation of claim 4 it is also necessary to show anticipation of '554 parent claims 1, 2 and 3.

265. The purpose of the system described in the reference is to allow dynamic output from various systems, including mathematical software, which are not themselves web-oriented, to be incorporated into dynamic web pages for display through a browser. "The handler being invoked should return the appropriate HTML or other type of document or it should generate such a document on the fly. Our CGI-handlers fall into the second category--an HTML document is generated and returned to the HTTP server." (p. 4) Thus the system produces dynamic ("on the fly) HTML documents (web pages) in response to a request and therefore meets preamble 1a.

266. The reference teaches a user at a browser interacting with an HTTP server (p. 5) and making a request for content. The HTTP server spawns a CGI handler, which passes the

request off to the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server. The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." (p. 5). Because ES is software that generates the dynamic page, it is the page server. Limitation 1b1 is met because the request is routed (sent) from the HTTP (web) server to the Exercise (page) server.

267. Limitation 1b2 is met because the Exercise Server cannot process the request unless it receives it.

268. Limitation 1b3 is met because the ES can be run on a separate computer from the HTTP server. The reference teaches using TCP/IP to pass the request to the Exercise Server, which is TCP releasing according to epicRealm's contentions. "Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5, para. 4 under "Implementation.).

269. Limitation 1b4 is met because the request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler, which functions as a dispatcher.

270. Limitation 1b5 is met because the request is sent from the HTTP(web) server to the CGI-handler (dispatcher).

271. Limitation 1b6 is met because the CGI-handler (dispatcher) dispatches the request to the Exercise (page) server based on the nature of the request. "The server side of interactive exercises (or more generally interactive documents) is implemented as an extension of a standard HTTP server. CGI is the standard for running such extension programs under HTTP servers [9]. Consequently, these programs are referred to as CGI-handlers (gateways and CGI-scripts are also in use). In UNIX, the HTTP-server spawns the handler which is referred to by the URL of an incoming information request." (p. 4, para. 1 under "Implementation.) If the request requires using Mathematica to generate content, then the CGI-handler knows to invoke the Exercise Server, which has Mathematica available. The CGI-Handler (dispatcher) makes an "informed selection" (i.e. does "dispatching") by extracting the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Therefore the reference teaches Single-Server Dispatching.

272. Under an alternative theory, the page server is the ES combined with the particular Mathematica session from which data is being retrieved. The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers.

273. Limitation 1c is met because the reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." (p. 4 para. 3 under "Implementation.") "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.

274. Limitation 1d1 is met because the system generates an HTML (web) page in response to the request. The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5.

275. Limitation 1d2 is met because real-time "interactive" content is inserted into the generated web pages: "The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients. All the plots created at runtime are converted to GIF files using UNIX utility programs and their URL(s) are embedded in the documents." (p. 5, para. 6 under "Implementation") The data sources from which data is retrieved are the Mathematica sessions.

276. The limitation of claim 2 is met because strings are used to identify the sources of interactive elements to be included in the generated HTML document: "The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings." (p.5, para. 6 under "Implementation.). The HTTP request which spawns the request to Mathematica includes a CGI request that "identifies" the Mathematica data source to be accessed. p. 5. "The first command explicitly refers to the exercise to be opened. It is an equivalent of the initial GET request. In response, inside Mathematica's environment, a session is initiated and a new context is devoted to it. Inside this context, the state of the session can be conveniently stored as values of symbols, safely isolated from other concurrent sessions. The name of the context is returned in a hidden field, which is part of each form which the generated document may contain. The second command corresponds to a POST request and always refers to a particular session by the name of its context." Further, the HTML templates provided to Exercise Server include strings generated by the CGI-Handler

that identify the sources of interactive elements to be included in the generated HTML document. p. 5.

277. The limitation of claim 3 is met because the required dynamic retrieval is performed when the data is retrieved from the sources identified by the strings to compose the final page: "The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." (p 5, para. 6 under "Implementation.)

278. The limitation of claim 4 is met because the reference expressly teaches maintaining information about each concurrent user: "In response, inside Mathematica's environment, a session is initiated and a new context is devoted to it. Inside this context, the state of the session can be conveniently stored as values of symbols, safely isolated from other concurrent sessions." This storing as "values of symbols" constitutes a connection cache. The Exercise Server (page server) maintains state information about the different Mathematica contexts in the Exercise Server's main memory: This "simplifies the task of maintaining the state information. It is no longer necessary to encode/decode data structures to/from flat representation at each HTTP request because it can be kept throughout the entire session in Exercise Server's main memory." p. 4.

279. Because reference 00Q incorporates all the limitations of claims 1, 2, 3 and 4, it anticipates at least those claims.

280. Claim 4 is also anticipated by reference 141, Derler, which teaches, "In contrast to the concept of handling every single WWW request as a separate Hyper-G session, the implementation of the new gateway described in this chapter is based on the idea of keeping up the connection between the gateway and the Hyper-G server during the processing of consecutive HTTP requests." Sec. 4.2.1. This maintaining of the connection constitutes a connection cache. This meets the limitation of claim 4.

281. Claim 4 is anticipated by the Oracle 1.0 References. Connection caching is used in SQL*Net, which may optionally be used with OWS 1.0. ORCL01806717.

282. Claim 4 is anticipated by at least 18 references, as listed in Exhibit 4.

'554 Claim 5

283. Claim 5 of the '554 patent is subsumed by claim 6 of the '335 patent. Claim 6 of '335 includes all the limitations of '554 claim 5. Any reference that anticipates '335 claim 6 also anticipates '554 claim 5. It is shown below that '335 claim 6 is anticipated by reference 00K. Therefore '554 claim 5 is also anticipated by reference 00K. Nevertheless, it will be shown that '335 claim 4 is anticipated by reference 180, Rogers et al. U.S. Patent 5,701,451. To show anticipation of claim 5 it is also necessary to show anticipation of '554 parent claims 1, 2 and 3. However, it was shown above that reference 180 already anticipates claims 1 and 2.

284. The reference teaches dynamically retrieving data from one or more data sources: "Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely…." 4:55-59. Therefore the reference anticipates claim 3.

285. The limitation of claim 5 is met because logging in to a data source is expressly taught in the reference. Step 115 of Fig. 8 shows a "DIS logon." It is described at 13:51-55: "At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS 'desktop', our DIS file server 14." The reference also teaches the use of passwords to obtain access to system functions: "It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any 'hidden' default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the 'hidden' password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet." (10:48-56). The password authorization is required to gain access to the data source.

286. Because reference 180 incorporates all the limitations of claims 1, 2, 3 and 5, it anticipates at least those claims.

287. Claim 5 is also anticipated by the Oracle 1.0 References because they teach the step of logging into one or more data sources. Reference 153 states, "In order to connect to an Oracle 7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (MAC003146). The username and password are used to log in. Reference 503 states on p. 4 that the Web Agent "[e]xtracts the database service name embedded in the URL [of the received request] and logs on to the database using the userid/password defined by the service.") Thus the Oracle 1.0 References teach the limitation of claim 5.

288. Claim 5 is anticipated by at least 15 references, as listed in Exhibit 4.

'554 Claim 6

289. Claim 6 of the '554 patent is subsumed by claim 7 of the '335 patent. Claim 7 of '335 includes all the limitations of '554 claim 6. Any reference that anticipates '335 claim 7 also anticipates '554 claim 6, which includes at least 00H, 00O, 016, 051, 141, 161, 501/502, 601, 604/605 and 607. It is shown expressly below that '335 claim 7 is anticipated by reference 002.

290. Claim 6 is also anticipated by Reference 153, one of the Oracle 1.0 References. Pages MAC003097-98, MAC003223-24 and MAC003255-56 of the reference explain how to set up file caching, which "decreases the access time required for commonly-used files." Thus the references teaches the limitation of claim 6.

291. Claim 6 is anticipated by at least 12 references, as listed in Exhibit 4.

'554 Claims 7 and 8

292. Claims 7 and 8 anticipated by reference 141, "The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services." It teaches generating dynamic web pages by sending a request for such a page to a "Hyper-G" server. In Hyper-G is a system developed at the University of Graz that allows any of a collection of distributed objects residing on different servers to be assembled in real-time to assemble a

dynamic web document: "When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items." (Sec. 3.3.4). The reference teaches that "[t]he server is free to generate requested documents by sending real files or by generating virtual hypertext on the fly" (Sec. 2.2) and thus deals with dynamic Web page generation requests. This meets the preamble 1a.

293. The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)." Sec. 4.2.3. Limitation 1b1 is met because the reference teaches routing (sending) a request from a master process (web server) to a Hyper-G server (page server): "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server), receives the response from the Hyper-G server, transforms the response again if necessary, sends it to the passing process, and closes the connection to the passing process."

294. Limitation 1b2 is met because the Hyper-G server (page server) must receive the request in order to process it. "Receiving" is expressly taught in Sec. 4.2.1: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session."

295. Limitation 1b3 is met because "The WWW side handles a transaction between WWW client and gateway according to the HTTP protocol." Sec. 4.2.1. Thus TCP releasing occurs according to epicRealm's contentions. Also, the reference expressly teaches "parallel processing of requests" in Sec. 4.2.2. It discusses the problem that "the master process would have to wait until the slave has sent the response in order to pass it to the WWW client. The master would not be able to accept further requests while others are being processed, which

returns us to the problem previously discussed," and offers a solution: "Therefore, the task of passing the request, waiting for the response, and finally returning the response to the client has to be performed by another process." This allows the master process (web server) to deal with other requests. The page server expressly communicates with its calling process: the slave process "opens a connection to the Hyper-G server, starts a session, reads the request from the WWW client using the connection inherited from its parent, and sends a response which is either a standard entry page containing a link with the new URL or a redirection message." (Sec. 4.2.2.) Hyper-G is implemented in Unix (Sec. 4.2.2), so the connection to the slave process involves SBL Releasing.

296. Limitation 1b4 is met because the master process (web server) does not process the request but spawns a slave process to deal with it. The handling of the request is intercepted at the web server and not processed by it.

297. Limitation 1b5 is met because the master process (web server) passes the request to a slave process (dispatcher), which sends it to a Hyper-G (page) server.

298. Limitation 1b6 is met because dispatching (sending) the request to the appropriate Hyper-G server is determined by Hyper-G's search facility, which locates the server on which the required information object resides (Secs. 3.3.1 and 3.5). "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." (Sec. 4.2.1)

299. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few

seconds for the completion of processing in some rare cases is acceptable for the user." (Sec. 4.2.2).

300. Limitation 1c is met because requests are processed in parallel. "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." (Sec. 4.2.2).

301. Limitations 1d1 and 1d2 are met because the Hyper-G (page) server assembles a dynamic web page using data retrieved from data sources. Sec. 4.1 refers to the "integration of WWW databases into Hyper-G."

302. Because reference 141 incorporates all the limitations of claim 1, it anticipates at least that claim.

303. The limitation of claim 2 is met because the Hyper-G server, in processing the request, identifies one or more data sources from which to retrieve data. Sec. 3.3.4. "The Hyper-G server is able to store 'remote objects' (i.e. Hyper-G objects in the link database that are pointers to remote systems) for accessing WWW, Gopher and WAIS servers. This allows Hyper-G clients to contact such remote systems and presenting a 'Hyper-G view' (as far as possible) of information on non-Hyper-G servers." Thus reference 141 anticipates claim 2.

304. The limitation of claim 2 is met because during the process where the Hyper-G system dynamically generates a web page, the Hyper-G server dynamically retrieves data from one or more data sources, i.e., the Hyper-G server's database or other data source. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"); Figure 4.13. Also Section 3.3.4. Thus reference 141 anticipates claim 3.

305. The limitation of claim 7 is met because HTML extensions are described in Sec. 4.6.1.1, where non-HTML "place holders" (the extensions) are added to documents. The placeholders are clearly not HTML because they begin and end with "%%", which is not part of HTML. Thus reference 141 anticipates claim 7.

306. Claim 7 is anticipated by at least 22 references, as listed in Exhibit 4.

307. The limitation of claim 8 is met because "place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1. Thus reference 141 anticipates claim 8.

308. Claim 8 is anticipated by reference 153, which is one of the Oracle 1.0 References. The reference teaches inserting dynamic data obtained by accessing an SQL database into an HTML extension template. Many examples of insertion are given. For example, the procedure htp.cite, described on page 6-20 of the reference (MAC003182), when called with two arguments, e.g. htp.cite(ctext, cattributes), causes the arguments to be substituted into a template as follows: <CITE cattributes>ctext</CITE>. Thus reference 153 teaches the limitation of claim 8.

309. Claim 8 is anticipated by at least 21 references, as listed in Exhibit 4.

'554 Claim 9

310. Claim 9 is anticipated by reference 141, Derler, "The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services." The reference teaches a networked method for managing a dynamic Web page generation request to a Web server, meeting preamble 9a. "When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items." Sec. 3.3.4.

311. Limitation 9b is met because the required "data sources" are databases in the reference.

312. Limitation 9c is met because the Web server is the WWW gateway of the reference. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3. The "page processing means" is the processor on which the page server software is running.

313. The Texas Court has construed "means for generating a request" to mean "a processor of a computer that is, or has, a Web client running a Web browser" or equivalents thereof. Any system in which requests are generated by a web browser includes such a means. Here, limitation 9d is met because the "first computer system" includes the client browser. "To meet this requirement, a gateway should be provided which can be used together with each Hyper-G server allowing WWW browsers to connect to Hyper-G." Sec. 4.1.

314. Limitation 9e1 is met because the "second computer system" includes the WWW gateway, which acts as web server.

315. The Texas Court has construed "means for receiving said request from said first computer" to mean "a processor of a computer that is, or has, a Web server running Web server executable" or equivalents thereof. Any system which has a Web server that receives dynamic page generation requests includes such a means. Here, limitation 9e2 is met because the second computer system includes a means for receiving the request ("WWW gateway") from the browser (first computer).

316. Limitation 9e3 is met because the second computer system includes a router to send the request to the Hyper-G Server (page server): "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)." Sec. 4.2.3

317. Limitation 9e4 is met because the handling of the request is intercepted by the Master Process of the Web server intercepts the handling of the request. The Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2.

318. Limitation 9e5 is met because the master process passes the request from the second computer system to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.

319. Limitation 9e6 is met because the selection of page server is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2.

320. Limitation 9e7 is met because receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1.

321. Limitation 9e8 is met because Figure 4.17 of the reference shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to epicRealm's contentions.

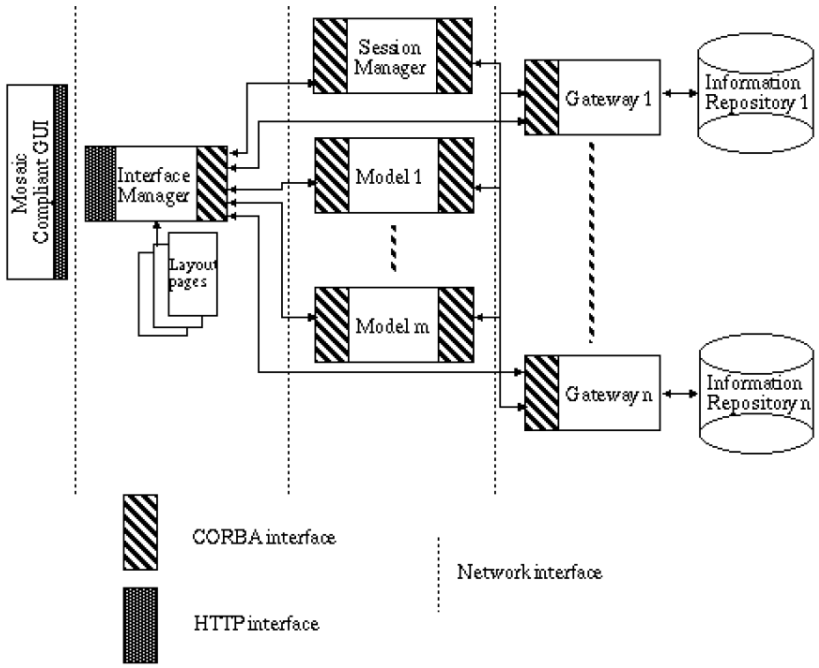
322. The Texas Court has construed "page server processing means" as "a processor of a computer that runs Page server software (wherein Page server software is page-generating software that generates a dynamic Web page)." (Markman Order p, 34.) Any computer system that has a page server has such a "page server processing means."

323. Limitation 9e9 is met because the Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13

324. Limitation 9e10 is met because data is accessed from databases. Sec. 4.1 teaches "integration of WWW databases into Hyper-G".

325. Because reference 141 incorporates all the limitations of claims 9, it anticipates at least claim 9.

326. Claim 9 is also anticipated by reference 019, Srinivas, "Applications of Mosaic in Health Care Delivery," which was published in October 1994 in connection with the Second International WWW Conference in Chicago. The reference discloses a system called ARTEMIS for accessing and updating databases of medical information through a WWW interface. A figure from the reference is shown below:



327. Patient data is stored in "heterogeneous, distributed databases" (p. 3), then retrieved on demand via queries and pages are converted on the fly to web pages using an HTML interface. Therefore preamble 9a is met.

328. Limitation 9b is met because the required "data sources" are databases of medical information in the "information repositories" accessed by gateways.

329. Limitation 9c is met because the "page server having a processing means" is shown in the figure as a model and/or gateway.

330. The Court has construed "means for generating a request" to mean "a processor of a computer that is, or has, a Web client running a Web browser" or equivalents thereof. Any system in which requests are generated by a web browser includes such a means. Here, limitation 9d is met because the "first computer system" includes the "Mosaic Compliant GUI" of the figure. Mosaic is a web browser.

331. Limitation 9e1 is met because the "second computer" includes the interface manager, session manager, and gateways, as shown in the figure.

332. The Court has construed "means for receiving said request from said first computer" to mean "a processor of a computer that is, or has, a Web server running Web server executable" or equivalents thereof. Any system which has a Web server that receives dynamic page generation requests includes such a means. Here, limitation 9e2 is met because the second computer system includes a means for receiving the request ("Interface Manager") from the Mosaic Compliant GUI (first computer).

333. Limitation 9e3 is met because the second computer includes a router (sender) to send the request to the "Session Manager," and "Gateways" (the page server).

334. Limitation 9e4 is met because the handling of the request is intercepted by Interface Manager, not handled by the Web server, but is sent to the page server for processing.

335. Limitation 9e5 is met because the Interface Manager (second computer) routes (sends) the request to the Session Manager, which acts as dispatcher.

336. Limitation 9e6 is met because the Session Manager (dispatcher) makes an informed selection of gateway to which it will send the request. "The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An

informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure, the Session Manager must maintain dynamic information about which Gateways 1 through n (page servers) are connected so it knows which Information Repositories 1 through n are available for responding to the request. This is described at page 10 of the reference: "A logical URL names a set of servers which contain the desired document, but does not refer to any particular physical server. When a request is sent to a logical URL, any server in the set may respond. The client is freed from any consideration of the physical server responding to the request, and servers can enter and leave the set without any awareness on the part of clients. This kind of behavior is required in high availability transaction processing systems." The import of this quotation is that to choose a suitable actual server given a logical URL, the system must maintain dynamic information about servers.

337. Limitation 9e7 is met because the Server (page server) receives the request.

338. Limitation 9e8 is met because the diagram shows that requests are communicated within the system through CORBA interfaces, which provide CORBA Releasing.

339. The Court has construed "page server processing means" as "a processor of a computer that runs Page server software (wherein Page server software is page-generating software that generates a dynamic Web page)." (Markman Order p, 34.) Any computer system that has a page server has such a "page server processing means."

340. Limitation 9e9 is met because ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."

341. Limitation 9e10 is met because data is accessed from databases called "Information Repositories" and inserted in the dynamic page. Page 5 of the reference shows an SQL query embedded in an HTML file. When the query is executed, data is retrieved and substituted for the query text.

342. Because reference 019 incorporates all the limitations of claims 9, it anticipates at least claim 9.

343. Claim 9 is further anticipated by the Oracle 1.0 References, which teach a networked system for managing a dynamic Web page generation request, meeting the preamble 9a of claim 9.

344. The Oracle 1.0 References teach one or more data sources. Reference 153 states on p. 1-4, "A Web Agent will connect to a single Oracle Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." (MAC003054). Thus the Oracle 1.0 References teach element 9b.

345. The Oracle 1.0 References teach a page server having a processing means. The page server is the Oracle7 Server. The processing means is the computer on which the Oracle Server executes. Thus the Oracle 1.0 References teach element 9c.

346. The Oracle 1.0 References teach a first computer system including means for generating a dynamic Web page generation request. That means is the computer running the User's web browser as shown on page 1-3 of reference 153 (MAC003053). Thus the Oracle 1.0 References teach element 9d.

347. The Oracle 1.0 References teach a second computer system including means for receiving a dynamic Web page generation request from said first computer. The second computer system is the web listener, which includes a means for receiving the request (web server). Thus the Oracle 1.0 References teach elements 9e1 and 9e2.

348. That the Oracle 1.0 References teach a router for routing the request from the second computer system to the page server was established above in connection with the discussion of the application of the Oracle 1.0 References to claim step '554 1b1. The router is that component of the web server which sends the request to the page server. Thus the Oracle 1.0 References teach element 9e3.

349. That the Oracle 1.0 References teach that routing further includes intercepting the request at the second computer was established above in connection with the discussion of claim step 1b4. The Web Listener, which runs on the second computer, intercepts the request. Thus the Oracle 1.0 References teach element 9e4.

350. That the Oracle 1.0 References teach routing the request from the second computer to a dispatcher was established above in connection with the discussion of claim step 1b5. Thus the Oracle 1.0 References teach limitation 9e5.

351. Element 9e6 is identical to element 1b6. It was established above that the Oracle 1.0 References teach this element.

352. The Oracle 1.0 References teach that the page server receiving the request. Otherwise, the page server would be unable to process the request. Thus the Oracle 1.0 References teach limitation 9e7.

353. It was established in connection with the discussion of claim step 1b3 that the Oracle 1.0 References teach releasing the Web server to process other requests. The second computer system includes the web listener (web server). Thus the Oracle 1.0 References teach limitation 9e8.

354. Limitations 9e9 and 9e10 taken together are the same as limitations 1d1 and 1d2. It has been shown above that the Oracle 1.0 References teach limitations 1d1 and 1d2. Therefore, the Oracle 1.0 References also teach l imitations 9e9 and 9e10.

355. Because the Oracle 1.0 References teach all the limitations of claim 9, they anticipate claim 9.

356. Claim 9 is anticipated by at least 29 references, as listed in Exhibit 4.

'554 Claim 10

357. Claim 10 adds the limitation to claim 9 "wherein said router in said second computer system includes: [10b] an interceptor intercepting said request at said second computer system and routing said request; and [10c] a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server."

358. Claim 10 is anticipated by Derler reference 141, Srinivas, "Applications of Mosaic in Health Care Delivery," which was shown immediately above to anticipate claim 9. The interceptor of element 10b is that component of the Master Process of the Web server which spawns a slave process to handle a dynamic page generation request. Sec. 4.2.2.

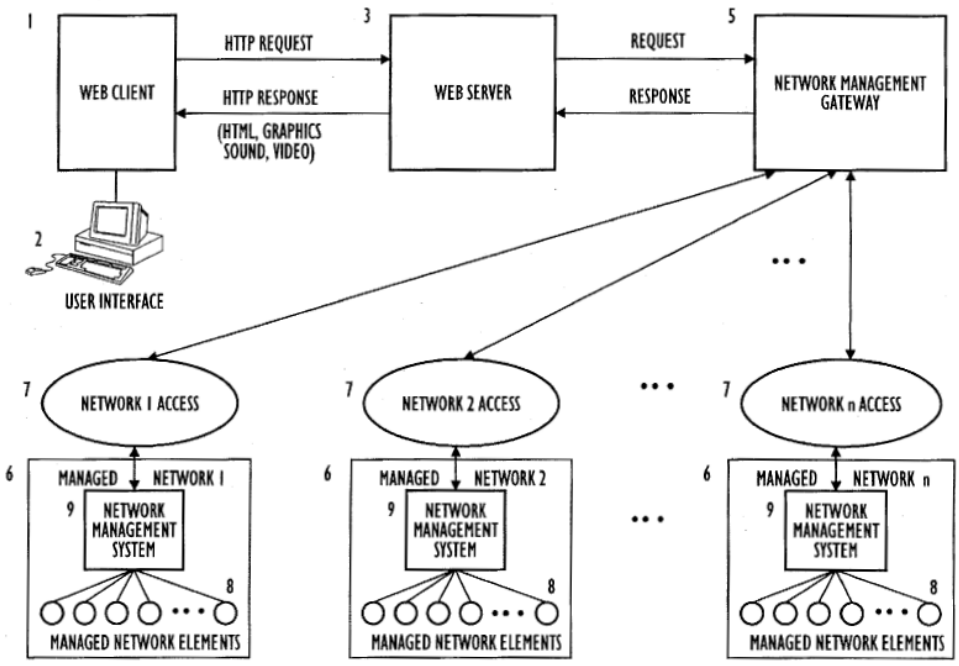
359. The dispatcher of element 10c is that component of the Master Process which spawns page servers, if necessary, and selects page servers based on dynamic information. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2

360. Claim 10 is also anticipated by reference 161, Scholl et al. U.S. Patent 5,742,762. The reference teaches a system for real-time (dynamic) construction of web content by accessing one or more "managed networks" to obtain information, which is then inserted into web pages for return to the requesting client. To show anticipation of claim 10, anticipation of claim 9 must first be demonstrated.

361. The Abstract of the reference states, "The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server, appearing to the client as information in a Web file." Thus it

teaches a networked system for managing a dynamic Web page generation request, as required by preamble 9a.

362. Figure 3 from the Scholl reference is reproduced below:



363. The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The "one or more data sources" are the managed network elements 8. Also Fig. 6 ("receive information from managed network").

364. The "page server having a processing means" of element 9c is performed by a combination of the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. The "processing means" is the computer on which systems 9 and output formatter 17 operate. Under an alternative theory based on epicRealm's contentions, the "page server having a processing means" is the network management system 9 of each managed network 6. Even though the network management system 9 does not produce finished pages, its output is displayable by a browser and hence, according to epicRealm, constitutes a web page. Therefore, network management system 9 can be considered a page server.

365. The "first computer system including means for generating said request" of element 9d is the web client 1 of Fig. 3 of the reference.

366. The "second computer system" of element 9e1 is web server 3.

367. The "means for receiving said request from said first computer" of element 9e2 is web server 3.

368. The "router … routing said request from said second computer system to said page server" of element 9e3 is a portion of network management gateway 5. Under the alternate theory under which network management system 9 is the page server, the router is also the network management gateway 5.

369. The "said routing further includes intercepting said request at said second computer system of element 9e4 occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." (2:42-44).

370. The "routing said request from said second computer to a dispatcher" of element 9e5 occurs when the web server sends the request to the network management gateway. (Arrow from box 3 to box 5 in the above figure.)

371. The "dispatching said request to said page server" of element 9e6 occurs in block 25 of Fig. 6 when the network management gateway determines which "appropriate network proxy agent" should receive the request and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen.

372. The "page server receiving said request" of element 9e7 occurs when the "appropriate network proxy agent" receives the request, which must occur if the request is to be processed.

373. The "releasing said second computer system to process other requests" of element 9e8 occurs when the network management gateway acknowledges receipt of the request to the web server. That the second computer system (web server 3 and network management gateway

5) is released is inherent in the architecture of the system disclosed in the reference. The reference teaches that "the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." (6:13-15). The mode of Internet communication taught in the reference is HTTP (Abstract), which provides acknowledgement of receipt, a form of releasing according to epicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server, as discussed above. Hence element/step 9e8 is taught. See also 8:42-49.

374. The "page server processing means processing said request and dynamically generating a Web page in response to said request" of element 9e9 occurs when the response to the request is converted "in real time to hypermedia document format in HTTP and HTML," as described in the Abstract.

375. The "said Web page including data dynamically retrieved from said one or more data sources" of element 9e10 occurs because information from the managed networks is incorporated in real-time into the generated page. Therefore the reference anticipates claim 9.

376. Claim 10 adds the limitation to claim 9 "wherein said router in said second computer system includes: [10b] an interceptor intercepting said request at said second computer system and routing said request; and [10c] a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server." In all of the applicable references the "interceptor" is a computer running code that performs the step of intercepting and the "dispatcher" is a computer running code that performs the step of dispatching.

377. The interceptor of element 10b is the software in the web server that determines that the request is not to be processed by the web server but is to be handled by the page server. The interceptor code is indicated by boxes 22 and 23 of Fig. 6 of the reference. The dispatcher of element 10c is the software corresponding to block 25 of Fig. 6, which determines which "appropriate network proxy agent" should receive the request and forwards the request to that agent.

378. Therefore, reference 161 anticipates claim 10.

379. Further, the Oracle 1.0 References anticipate claim 10. Preamble 10a is met because the Oracle 1.0 References anticipate claim 9.

380. The interceptor element 10b is that component of the web listener which detects that the request is not for a static page, but for a dynamic one. The interceptor routes the request to the Oracle7 Server through the web agent. Thus the Oracle 1.0 References teach element 10b.

381. The dispatcher element 10c is that component of the web agent that dispatches the request to the Oracle7 server. Thus the Oracle 1.0 References teach element 10c.

382. Because the Oracle 1.0 References teach all the elements of claim 10, they anticipate claim 10.

383. Claim 10 is anticipated by at least 29 references, as listed in Exhibit 4.

'554 Claim 11

384. Claim 11 is anticipated by reference 141, "The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services." It teaches generating dynamic web pages by sending a request for such a page to a "Hyper-G" server. In Hyper-G is a system developed at the University of Graz that allows any of a collection of distributed objects residing on different servers to be assembled in real-time to assemble a dynamic web document: "When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4.) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items." (Sec. 3.3.4)

385. The preamble 11a is met because the reference discloses a software system. A software system must reside on a machine-readable medium with stored computer instructions.

386. Limitation 11b1 is met because the reference teaches that "[t]he server is free to generate requested documents by sending real files or by generating virtual hypertext on the fly" (Sec. 2.2) and thus deals with dynamic Web page generation requests. The reference teaches routing (sending) a request from Sec. 4.2.3 describes how a request received by a master process (web server) is sent to a Hyper-G server (page server): "When it has accepted a connection

initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server), receives the response from the Hyper-G server, transforms the response again if necessary, sends it to the passing process, and closes the connection to the passing process."

387. Limitation 11b2 is met because the Hyper-G server (page server) must receive the request in order to process it. "Receiving" is expressly taught in Sec. 4.2.1: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session."

388. Limitation 11b3 is met because "The WWW side handles a transaction between WWW client and gateway according to the HTTP protocol." Sec. 4.2.1. Thus TCP releasing occurs according to epicRealm's contentions. Also, the reference expressly teaches "parallel processing of requests" in Sec. 4.2.2. It discusses the problem that "the master process would have to wait until the slave has sent the response in order to pass it to the WWW client. The master would not be able to accept further requests while others are being processed, which returns us to the problem previously discussed," and offers a solution: "Therefore, the task of passing the request, waiting for the response, and finally returning the response to the client has to be performed by another process." This allows the master process (web server) to deal with other requests. The page server expressly communicates with its calling process: the slave process "opens a connection to the Hyper-G server, starts a session, reads the request from the WWW client using the connection inherited from its parent, and sends a response which is either a standard entry page containing a link with the new URL or a redirection message." (Sec. 4.2.2.) Hyper-G is implemented in Unix (Sec. 4.2.2), so the connection to the slave process involves SBL Releasing.

389. Limitation 11b4 is met because the master process (web server) does not process the request but spawns a slave process to deal with it. Thus the request is intercepted (deflected away from the web server).

390. Limitation 11b5 is met because the master process (web server) passes the request to a slave process (dispatcher), which sends it to a Hyper-G (page) server.

391. Limitation 11b6 is met because dispatching (sending) the request to the appropriate Hyper-G server is determined by Hyper-G's search facility, which locates the server on which the required information object resides (Secs. 3.3.1 and 3.5). "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." (Sec. 4.2.1)

392. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." (Sec. 4.2.2).

393. Limitation 11c is met because requests are processed in parallel. "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." (Sec. 4.2.2).

394. Limitation 11d is met because the Hyper-G (page) server assembles a dynamic web page using data retrieved from data sources. Sec. 4.1 refers to the "integration of WWW databases into Hyper-G."

395. Because reference 141 incorporates all the limitations of claim 11, it anticipates at least claim 11.

396. Claim 11 is further anticipated by the Oracle 1.0 References. Claim 11 is drawn to a machine-readable medium containing a computer program which performs the steps of claim 1. Because Oracle WebServer as described in the Oracle 1.0 References performs all the steps of claim 1, claim 11 reads on the machine-readable medium on which Oracle 1.0 Server resides.

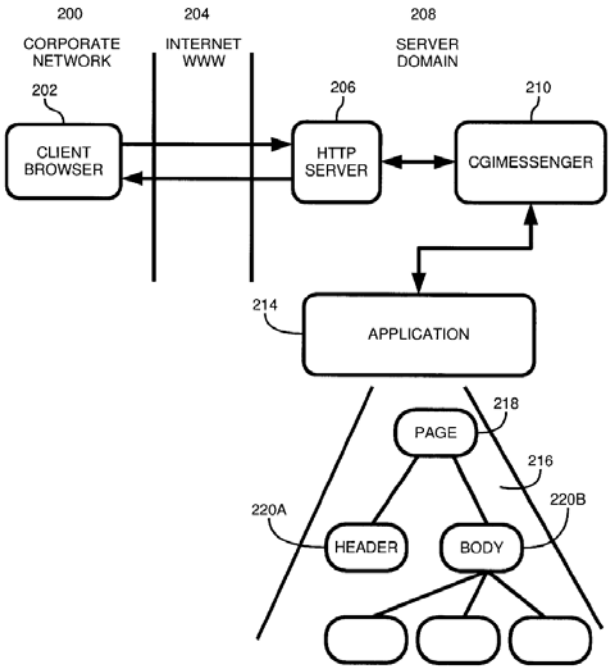
397. Claim 11 is anticipated by at least 29 references, as listed in Exhibit 4.

'335 Claim 1

398. Claim 1 of '335 is subsumed by claim 1 of '554. Every step of '335 claim 1 is present in '554 claim 1. The claims differ in that steps 1b5 (routing to a dispatcher) and 1b6 (dispatching) of '554 claim 1 are not present in '335 claim 1. Therefore, any reference that anticipates '554 claim 1 also anticipates '335 claim 1, so it has already been shown that '335 claim 1 is anticipated by references 00Q, 063, 141, 180 and the Oracle 1.0 References. Nevertheless, it will be shown that '335 claim 1 is anticipated by reference 194, Popp et al. U.S. Patent 6,651,108, filed August 14, 1995.

399. The reference teaches processing dynamic page generation requests: "The object-oriented approach used by the present invention provides the ability to dynamically generate a page, or HTML document. Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:63-6:1. This meets the preamble 1a.

400. Figure 2 of the reference is shown below. It is a three-tier architecture. Tier 1 is the client browser 202. Tier 2 includes the HTTP Server and the CGI messenger. Tier 3 is the application (page server).



401. The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in the figure. See also 6:25-7:25. This meets limitation 1b1

402. The page server must receive the request in order to process it. Receiving is shown in the figure as the double-headed arrow from box 210 to box 214. See also 6:25-7:25. This limitation 1b2 is met.

403. The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to epicRealm's contention. See 6:38-7:13. This meets limitation 1b3.

404. Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24. This meets limitation 1c1.

405. Limitation 1c2 is met because the request is routed to the page server: "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21.

406. Reference 194 incorporates reference 195 by reference. Reference 195 teaches running the web server and page server on different machines, in which case concurrent

processing is inherent: "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." (Popp '291 patent, 7:25-35). This meets limitation 1d.

407. The application 214 returns "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." 7:44-58. This meets limitation 1e1.

408. Limitation 1e2 is met because the references teaches that "[t]he contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318)." 8:44-46. Because reference 194 teaches all the limitations of claim 1, it anticipates claim 1.

409. Claim 1 is anticipated by at least 30 references, as listed in Exhibit 4

'335 Claims 2-4

410. Claim 2 of the '335 patent is virtually identical to claim 1 of the '554 patent. Thus, any reference that anticipates one anticipates the other. Therefore, it has already been shown that '335 claim 2 is anticipated by at least the '554 claim 1 anticipating references 00Q, 063, 141, 153, 180 and 501/502. Nevertheless, an additional reference will be used to demonstrate anticipation of '335 claim 2.

411. Claims 2-4 are anticipated by reference 194, Popp et al. U.S. Patent 6,651,108, which has already been shown to anticipate parent claim 1.

412. Limitation 2a is essentially a preamble, which is satisfied because the reference meets all the limitations of claim 1 and describes a computer-implemented method.

413. Limitation 2b is met because the dispatcher is the CGIMessenger 210 of Fig. 2 of the reference. The CGIMessenger chooses an application (page server) to which the request is forwarded: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.

414. Limitation 2c is met because the reference teaches informed selection based on dynamic information maintained about page servers: "For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers.

415. Claim 2 is anticipated by at least 29 references, as listed in Exhibit 4

416. Claim 3 adds the limitation to claim 1 "wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data." Claim 3 of the '335 patent is virtually identical to claim 2 of the '554 patent. Any reference that anticipates one anticipates the other. Therefore, it has already been shown that references 002, 00K, 00Q, 016, 066, 141, 161, 180, 501/502 and the Oracle 1.0 References, which anticipate '554 claim 2, also anticipate '335 claim 3. Nevertheless, it will be shown that reference 194 also anticipates '335 claim 3.

417. The data sources are the WAIS servers. Since Discover is a "single point of access to over 500 WAIS servers," it must identify one or more of them from which to retrieve data. This is done by the content router, using its routing database and refinement database, which is described in the "Content Labels" section of the reference. Therefore, the limitation of claim 3 is met and claim 3 is anticipated by reference 154.

418. Claim 4 adds the limitation to claim 1 "wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources." Claim 4 of the '335 patent includes all the limitations of claim 3 of the '554 patent. Any reference that anticipates one anticipates the other. Therefore, it has already been shown that reference 002, 00K, 00Q, 016, 066, 141, 153, 161, 180 and the Oracle 1.0 References, which anticipate '554 claim 4, also anticipate '335 claim 4. Nevertheless, it will be shown that reference 00W/063 also anticipates '335 claim 4.

419. To show anticipation of claim 4, anticipation of parent claims 1 and 2 must be shown. It was demonstrated earlier that the reference anticipates '554 claim 1. Every step of '335 claims 1 and 2 are also steps of '554 claim 1. Therefore the reference anticipates ''335 claims 1 and 2.

420. Reference 00W meets the limitation of '335 claim 4 as follows: "A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6. The generated web page therefore contains data dynamically retrieved fro one or more data sources and the reference anticipates claim 4.

'335 Claim 5

421. Claim 5 adds the limitation to claim 1 "wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources." Claim 5 is anticipated by reference 141 which has already been shown above to anticipate claim 1.

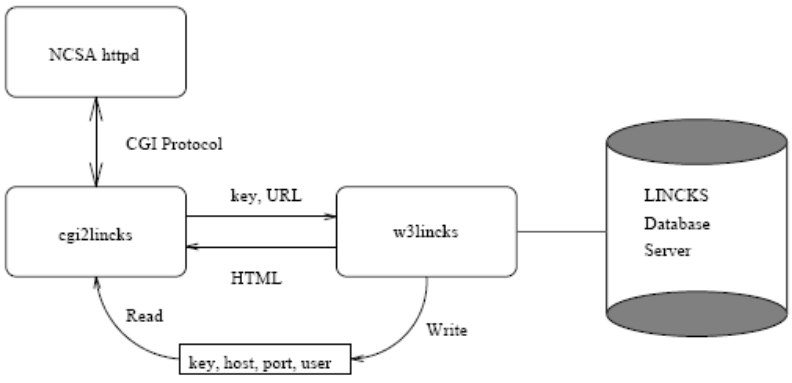
422. The reference states, “In contrast to the concept of handling every single WWW request as a separate Hyper-G session, the implementation of the new gateway described in this chapter is based on the idea of keeping up the connection between the gateway and the Hyper-G server during the processing of consecutive HTTP requests." Sec. 4.2.1. This maintaining of the connection constitutes a connection cache.

423. Claim 5 is also anticipated by the Oracle 1.0 References. It was shown above that reference 153 anticipates ’335 claim 1. Connection caching is used in SQL*Net, which may optionally be used with OWS 1.0. ORCL01806717. Therefore, the Oracle 1.0 References anticipate claim 5.

’335 Claim 6

424. Claim 6 is anticipated by reference 00K, Sjölin, “A WWW Front End to an OODBMS,” published in connection with the Second International WWW Conference in Oct. 1994. To show anticipation of claim 6 it is also necessary to show anticipation of ’335 parent claim 1.

425. The reference describes a system called LINCKS for obtaining real-time access to object-oriented database systems through the World Wide Web. Figure 6 from the reference is shown below.



426. The abstract of the reference states, “The prototype generates HTML documents on the fly from information pieces in the database.” Thus it is a computer-implemented method for

managing a dynamic Web page generation request to a Web server (NCSA httpd), meeting the preamble 1a.

427. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to the partially generated page and the w3lincks blocks in the diagram, which shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks, which is step 1b1.

428. The page server must receive the request or it could not be serviced, so step 1b2 is performed.

429. The web server (NCSA httpd) is shown as using the "CGI protocol" to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to epicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. "To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing. Therefore, the page server releases the web server and step 1b3 is performed.

430. The request for a dynamic page is intercepted and not processed by the web server, but is routed to the page server. Therefore, steps 1c1 and 1c2 are performed.

431. That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture shown in the diagram. Section 4.1 of the reference teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the

processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing. Therefore, step 1d is performed.

432. A dynamic "on the fly" web page is created, so step 1e1 is performed.

433. The dynamic web page includes data dynamically retrieved from the LINCKS Database Server, so step 1e2 is performed. Therefore, the reference anticipates claim 1.

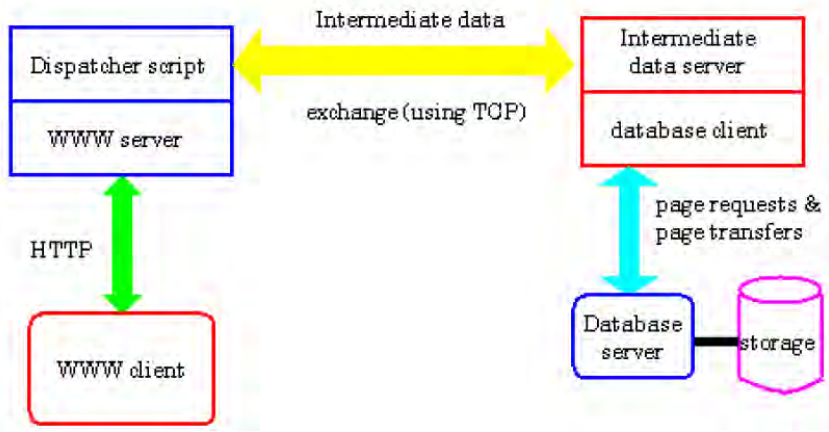
434. Claim 6 adds the limitation to claim 1 "wherein said step of processing said request includes the step of logging into said one or more data sources." Section 4.2 of the reference recites that "When starting w3lincks, it connects to a LINCKS database and login as a normal user." Thus the step of logging into said one or more data sources is performed and the reference anticipates claim 6.

435. Claim 6 is also anticipated by the Oracle 1.0 References. Any reference that anticipates '335 claim 1 and also teaches the step of logging into one or more data sources as part of processing the request anticipates claim 6. "In order to connect to an Oracle 7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (MAC003146). The username and password are used to log in to the data source: "A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." (p. 1-4: MAC 3054; ORCL 23). "When the Oracle Web Agent logs into the Oracle7 Server, it starts a PL/SQL procedure that has been created by the user to generate an HTML page as its output." (p. 1-4: MAC 3054; ORCL 23). Therefore, the Oracle 1.0 References teach logging into data sources and anticipate claim 6.

'335 Claim 7

436. Claim 7 is anticipated by reference 002, Varela, "DB: Browsing Object-Oriented Databases over the Web," published in December 1995 in connection with the Fourth Annual

WWW Conference held in Boston. It describes a system called DB (Database Browser) that generates dynamic web pages based on database queries. The Section "Databases in the Web" states, "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers. For such instances, there are scripts, which are programs that perform conversions from different data formats into HTML on-the-fly." Figure 2 from the reference is shown below. To show anticipation of claim 7 it is also necessary to show anticipation of '335 parent claim 1



437. The reference discloses a computer-implemented method for managing a dynamic Web page generation request to a Web server (WWW server in the diagram), which meets the preamble 1a.

438. The page server is shown in the diagram as the combination of the "Dispatcher script" and the "Intermediate data server." Page 7 of the references states, "The [Intermediate data] server performs the database operation and sends back the results using the same connection to the dispatcher script. The dispatcher script in turn parses the results and converts them to HTML. They are then forwarded by the Web server to the browser." Page 2 of the reference refers to a "page server architecture." Therefore, step 1b1, routing a dynamic Web page generation request from a Web server to a page server, is taught in the reference, as shown in the figure.

439. The page server must receive the request or it could not be serviced, so step 1b2 is performed.

440. The web server hands off the request to the Intermediate data server using TCP, as shown in the diagram. TCP provides acknowledgment of receipt, which is form of releasing, according to epicRealm. Therefore, the page server releases the web server and step 1b3 is taught in the reference.

441. The request for a dynamic page is intercepted and not processed by the web server, but is routed to the page server. Intercepting is performed by Dispatcher script, a CGI-script which is an extension of the WWW server (Web server) that intercepts handling of the request at the WWW server (Web server) by determining the type of query and sending the query to the database server for processing. Therefore, steps 1c1 and 1c2 are taught.

442. That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests. Therefore step 1d is found in the reference.

443. Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on the fly," according to the reference, so step 1e1 is performed.

444. The dynamic web page includes data dynamically retrieved from the Objectstore object-oriented database management system on the Database server, so step 1e2 is performed. Therefore, the reference anticipates claim 1.

445. Under an alternate theory, the dispatcher is the "Dispatcher script" and the "Intermediate data server" is a Partial Page Server under epicRealm's contentions.

446. Claim 7 adds the limitation to claim 1 "wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page." The last paragraph of the "Performance Evaluation" section of the reference states, "We also observed that closing the database does not actually result in a subsequent request taking a 'cold

time.' This result, we concluded, was because ObjectStore has a page server model which means that even when the database is closed, the data pages are still in memory and are not reclaimed unless the data of some other application is paged into the same place." This is page caching. The generated pages are stored in memory and, if needed, do not have to be generated anew. Thus the step of claim 7 is performed and the reference anticipates claim 7.

447. Claim 7 is also anticipated by the Oracle 1.0 References. Any reference that anticipates '335 claim 1 and also teaches the step of maintaining a page cache containing the dynamically generated web page anticipates claim 7. It was shown in connection with the discussion of '554 claim 6 that the Oracle 1.0 References teach this step. Therefore, the Oracle 1.0 References anticipate claim 7.

'335 Claim 8

448. Claim 8 is anticipated by reference 141, Derler, "The World-Wide Web Gateway to Hyper-G," which was shown above to anticipate parent claim 1. The reference teaches tag-based text templates in Sec. 4.6.1.1.

449. Claim 8 is also anticipated by the Oracle 1.0 References. Any reference that anticipates '335 claim 1 and also teaches that the page server includes tag-based text templates for configuring the web page anticipates claim 8. It was shown in connection with the discussion of '554 claim 7 that the Oracle 1.0 References teach use of custom HTML extension templates for configuring the web page. HTML templates are tag-based text templates. Therefore, the Oracle 1.0 References anticipate claim 8.

'335 Claim 9

450. Claim 9 adds the limitation to claim 8 "wherein said step of processing said request further includes the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates." Claim 9 is also anticipated by reference 141, which teaches, "The following place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1. Thus reference 141 anticipates claim 9.

451. Claim 9 is also anticipated by the Oracle 1.0 References. Any reference that anticipates '335 claim 8 and also teaches the step of inserting dynamically retrieved data from said one or more data sources into tag-based text templates anticipates claim 9. It was shown above that the Oracle 1.0 References anticipates '554 claim 8, which is a special case of '335 claim 9. (All HTML extension templates are tag-based text templates.) Therefore, the Oracle 1.0 References anticipate '335 claim 9.

'335 Claim 10

452. Claim 10 adds the limitation to claim 8 "wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request." Claim 10 is anticipated by reference 194, Popp et al. U.S. Patent 6,651,108.

453. The reference was shown above to anticipate '335 parent claim 1. It anticipates parent claim 8 because it teaches HTML templates, which are tag-based. "At step 302, a Web page developer creates the HTML templates to provide a Web page definition." 8:44-45. "When the Web page definition is requested by a client browser, the associated HTML template(s), script(s), and declarations are retrieved from storage. At step 306, a parser is invoked to parse the statements contained in the HTML template(s)." 8:54-58. "The present invention provides an extension to HTML that is used on the HTTP server side." 10:47-48. "The present invention uses an extension to the standard HTML known as the group extension The group extension provides the ability to combine HTML elements or statements in a single block." 10:57-607:33-62.

454. The reference also teaches the limitation of claim 10: "A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." 2:26-28. Thus the tag-based (HTML) text templates drive a format of the data and the reference anticipates claim 10.

455. Claim 10 is further anticipated by the Oracle 1.0 References, which have already been shown to anticipate claim 8. Section 5 of reference 153 (e.g., page 5-5, MAC003147)

describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. Therefore, the Oracle 1.0 References anticipate claim 10.

'335 Claim 11

456. Claim 11 adds the limitation to claim 8 "wherein said tag-based text templates include HTML templates." This limitation is perforce taught by reference 194 and the Oracle 1.0 References because the tag-based text templates taught therein are HTML templates. Likewise for the Oracle 1.0 References. Therefore, the Microsoft References and the Oracle 1.0 References anticipate claim 11.

'335 Claim 12

457. Claim 12 is anticipated by reference 180, Rogers et al. U.S. Patent 5,701,451, described above with reference to '554 claim 2. To show anticipation of claim 8 it is also necessary to show anticipation of '335 parent claim 1. However, such anticipation has already been shown above in the discussion of that claim.

458. Claim 12 adds the limitation to claim 1 "wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources." This limitation is taught in the reference at: 13:30-33: "This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retrie[ve]d from a database to be performed." It is also taught at 16: 2-4: "Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved." Because this limitation and all those of parent claim 1 are taught in reference 180, that reference anticipates claim 12.

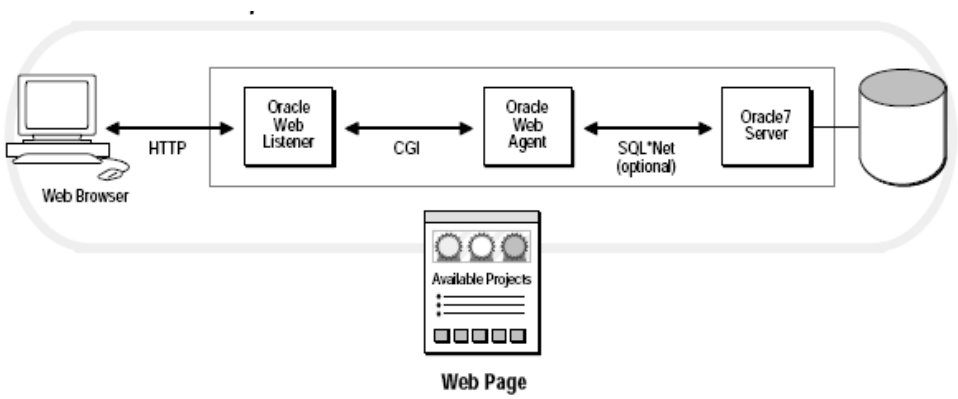
459. Claim 12 is also anticipated by the Oracles References. The Oracle 1.0 References have already been shown to anticipate claim 1. Dynamically updating the database is described on page 4-9 of reference 153 (MAC003113). Therefore, the Oracle 1.0 References anticipate claim 12.

'335 Claims 13 and 14

460. Claims 13 and 14 are anticipated by reference 173, Knudson, "Application Development with Microsoft's Internet Information Server," dated February 2, 1996, as shown in Exhibit 3..

'335 Claim 15

461. Claim 15 is anticipated by reference 153, Oracle Web Server User's Guide (Release 1.0). It teaches a system for generating dynamic web pages by passing requests off from an Oracle Web Listener (web server) to an Oracle Web Agent (dispatcher), which routes them to Oracle7 server (page server), which has access to dynamic data through an SQL interface to an Oracle database. A figure from page 1-3 of the reference is shown below.



462. The preamble 15a is met because the reference teaches a computer-implemented method.

463. Limitation 15b1 is met because the Oracle Web Listener (web server) is an HTTP-compliant device. Page 1-3 recites that "The Oracle Web Listener is a commercial-quality HTTP server that services document requests from any web browser. The Web Listener is shown as passing the request for a page off to the Oracle Web Agent (dispatcher), which routes it to the Oracle7 Server (page server). "The Oracle Web Agent handles requests for dynamic (program-created) pages. It handles the connection to an Oracle 7 Server, invocation of the requested procedure, and transmission of the resulting HTML document back to the Web browser." (1-2).

464. Limitation 15b2 is met because the Oracle7 Server (page server) cannot process a request unless it receives it. Receipt is shown in the diagram

465. Limitation 15b3 is met because the request is passed from the Web Listener (HTTP-compliant device) to the Oracle7 Server (page server) using HTTP, so TCP releasing is performed, according to epicRealm's contention. Page 3-2 recites that the Web Listener "is designed to run as an asynchronous engine," which means that it continues to respond to other requests after passing one of them off to the page server.

466. Limitation 15c1 is met because the request is intercepted (not processed) by the Web Listener (HTTP-compliant device) and sent (transferred) to Oracle7 Server (page server): "When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception.

467. Limitation 15c2 is met because the request is transferred to Oracle7 Server. "The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the Patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram.

468. Limitation 15d is met because the request is processed by Oracle7 Server to yield a dynamic web page. When the request is offloaded from the Web Listener, it is free to listen for, and process, other ("asynchronous") requests. The reference expressly discusses the "high performance" nature of the Web Listener: "For maximum performance the Web Listener is designed to run as an asynchronous engine via a single process with a single thread, in comparison to most HTTP servers, which start a new thread or process whenever a new connection is made. This considerably decreases the time required to service a request and

reduces Listener machine resource utilization, providing high performance under a heavy load." (Page 3-2). Because is runs a single thread, the Web Listener is free to process other requests once it has handed off a request to the Web Agent.

469. Limitations 15e1 and 15e2 is met because the very point of the reference is to include data retrieved from Oracle databases into dynamic web pages: "With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." (Page 5-5)

470. Therefore, claim 15 is anticipated by reference 153.

471. Claim 15 is further anticipated by the Oracle 1.0 References, which teach a computer-implemented method (preamble 15a).

472. In the Oracle 1.0 References, the HTTP-compliant device is the web listener, which transfers a dynamic web page generation request to a page server, which receives the request, as discussed in connection with '554 claim steps 1b1 and 1b2, above. Therefore, the Oracle 1.0 References teach steps 15b1 and 15b2.

473. The Oracle 1.0 References teach releasing the web listener (HTTP-compliant device) to process other requests (15b3), as discussed in above in connection with '554 step 1b3.

474. The Oracle 1.0 References teach intercepting the request at the web listener (HTTP-compliant device) and transferring the request to the Oracle7 Server (page server) (step 15c), as discussed above in connection with '554 steps 1b4 and 1b6.

475. Steps 15d1 and 15d2 correspond to '554 step 1c, which is taught by the Oracle 1.0 References.

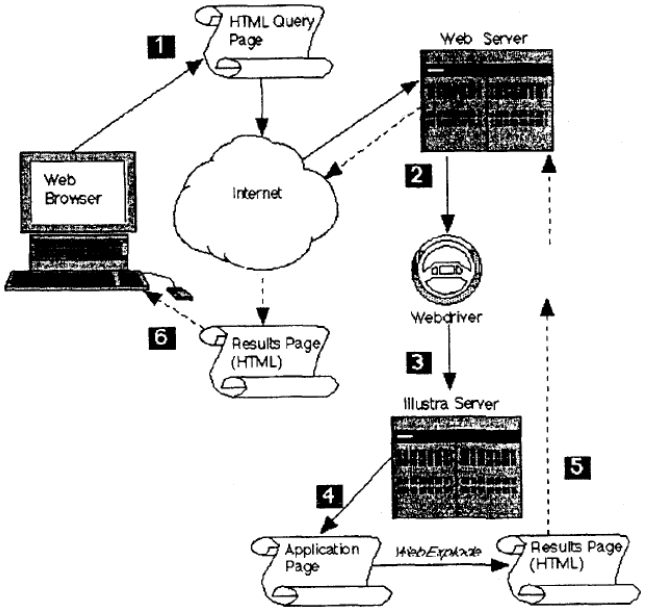
476. Steps 15e corresponds to '554 steps 1d1 and 1d2, which are taught by the Oracle 1.0 References. Therefore, all the steps of claim 15 are taught by the Oracle 1.0 References.

477. Claim 15 is anticipated by at least 30 references, as listed in Exhibit 4.

'335 Claim 16

478. Claims 16-20 are anticipated by reference 016, Gaffney, "Illustra's Web DataBlade Model." To show anticipation of these claims it is first necessary to show anticipation of parent claim 15.

479. The reference teaches a system for "creating Web-enabled applications that dynamically retrieve and update Illustra database content." (Abstract). Fig. 2 of the reference is reproduced below.



480. The method taught in the reference is computer-implemented, which meets preamble 15a.

481. The "HTTP-compliant device" of the claim is Web Server of the diagram. The page server of the claim is the Illustra Server. The diagram shows a request being transferred from the Web Server to the Illustra Server, which is step 15b1.

482. The Illustra Server receives the request or it could not perform step 5, so step 15b2 is performed.

483. That the Illustra Server releases the HTTP-compliant device to process other requests is inherent in the diagrammed architecture. The reference further teaches using

Webclient, which "optimizes connection management between the Web server and the Illustra database,"(p. 109) the stated purpose of which is to promote efficiency. The last paragraph on p. 109 refers to "requests to the database." If the HTTP-compliant device were not released to process other requests, there could never be more than one such request pending at any time. Page 112 teaches that Illustra DataBlade "works with all standard HTTP and standard Web servers," which means that it uses HTTP and thus accomplishes TCP releasing, according to epicRealm's contention. Therefore, step 15b3 is taught in the reference.

484. The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server. Therefore, the reference teaches steps 15c1 and 15c2.

485. Concurrency was discussed above in connection with step 15b3. Therefore, step 15d is taught in the reference.

486. The request is processed, or the Results Page could not be generated. Therefore, the reference teaches step 15e1.

487. The Results Page is dynamically generated in response to the request, Page 107 of the reference states, "Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted." The resulting page includes data retrieved by query from a database (Item 4, p. 110). Therefore, the reference teaches step 15e2, so reference 016 anticipates claim 15.

488. Claim 16 adds the limitation to claim 15 "wherein said step of transferring said request includes the steps of: [16b] transferring said request from said HTTP-compliant device to a dispatcher; and [16c] dispatching said request to said page server."

489. The dispatcher of the claim is the Webdriver shown in the diagram.

490. The Webdriver "connects to the specified database as a specified user." p. 110. Thus the Webdriver (dispatcher) makes an informed selection and performs dispatching within the meaning of the claim. The selection is based on dynamic information maintained about page servers, i.e., the database specification and dynamic load (load-balancing) data: "The Web

DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109. Therefore, the reference teaches steps 16b and 16c and anticipates claim 16.

491. Claim 16 is also anticipated by the Oracle 1.0 References, shown above to anticipate parent claim 15.

492. Preamble 16a is trivially satisfied because Oracle WebServer is computer-implemented. Limitations 16b and 16c are substantially identical to limitations '554 1b5 and '554 1b6, which were shown earlier to be met by the Oracle 1.0 References. Therefore, they anticipate claim 16.

493. Claim 16 is anticipated by at least 29 references, as listed in Exhibit 4.

494. Claim 17 adds the limitation to claim 15 "wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data." As explained above, the dispatcher identifies which database to access based on user specification. Therefore, reference 016 teaches the limitation of claim 17 and anticipates claim 17.

495. Claim 17 is also anticipated by the Oracle 1.0 References. Claim 17 is identical to '554 claim 2 except that it refers to a different parent claim. The Oracles 1.0 References have already been shown to teach the limitation of '554 claim 2. Therefore, the Microsoft References and Oracles References teach all the limitations of claim 17.

'335 Claim 18

496. Claim 18 adds the limitation to claim 15 "wherein said step of dynamically generating said page includes the step of dynamically retrieving said data from said one or more data sources." Item 4 on p. 110 of references 016 states, "WebExplode parses the extracted pages, executes the embedded queries, and then formats the results. This is accomplished within the database server." Thus the data is retrieved dynamically by queries and inserted into the generated page, which meets the limitation of claim 18. Therefore, reference 016 anticipates claim 18.

497. Claim 18 is also anticipated by the Oracle 1.0 References. Claim 18 is identical to '554 claim 3 except that it refers to a different parent claim. The Microsoft References and the Oracle 1.0 References have already been shown to teach the limitation of '554 claim 3. Therefore, the Oracle 1.0 References teach all the limitations of claim 18.

'335 Claim 19

498. Claim 19 adds the limitation to claim 15 "wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources." Page 110 of reference 016 states, "Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." Thus the system maintains information about "already-established connections," that is, a connection cache. The stated purpose of the cache is the same as the teaching of the '335 patent, namely that "Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. 4, Page server 404(1) can retrieve data from data source 406, data source 408 or data source 410. Page server 404(1) can maintain connection cache 412(1), containing connections to each of data source 406, data source 408 and data source 410, thus eliminating connect times from the Page servers to those data sources."

(6:57-64). Thus the limitation of claim 19 is taught in the reference and reference 016 anticipates claim 19.

499. Claim 19 is also anticipated by the Oracle 1.0 References. Claim 19 is identical to '554 claim 4 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '554 claim 4. Therefore, the Oracle 1.0 References teach all the limitations of claim 19.

'335 Claim 20

500. Claim 20 adds the limitation to claim 15 "wherein said step of processing said request includes the step of logging into said one or more data sources." Item 3 on p. 110 of reference 016 states, "the driver connects to the specified database as a specified user." Connecting as a specified user means logging in. Thus the reference teaches the limitation of claim 20, so reference 016 anticipates claim 20.

501. Claim 20 is also anticipated by the Oracle 1.0 References. Claim 20 is identical to '554 claim 5 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '554 claim 5. Thus the Oracle 1.0 References teach all the limitations of claim 20 and therefore anticipate it.

'335 Claims 21-25

502. Claims 21-25 are anticipated by reference 141, Derler "The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services." It must first be shown to anticipate '335 parent claim 15. It was already shown that Derler anticipates '554 claims 2 and 3.

503. Hyper-G is a computer-implemented method of generating dynamic web pages. Each step of claim 15 is a step previously shown to have been present in a claim anticipated by Derler as follows: 15b1 is analogous to '554 1b1; 15b2 is analogous to '554 1b2; 15b3 is analogous to '554 1b3; 15c1 is analogous to '554 1b4; 15c2 is subsumed within '554 1b1; 15d is analogous to '554 1c; 15e1 is analogous to '554 1d1; and 15e2 is analogous to '554 1d2. Therefore, Derler anticipates '335 claim 15.

504. Claim 21 is anticipated because Derler teaches page caching: "[I]n the Hyper-G system, requests from clients rather than being directly sent to document servers, are routed through a document cache. … If the document requested has previously been cached and can be found in the cache server, it is transmitted directly. Otherwise, the cache server retrieves the document from the document server, transmits it to the client and stores a copy of it in the cache." Sec. 3.3.5.

505. Claim 21 is also anticipated by the Oracle 1.0 References. Claim 21 is identical to '554 claim 6 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '554 claim 6. Therefore, the Oracle 1.0 References teach all the limitations of claim 21.

506. Claim 22 adds the limitation to claim 15 "wherein said page server includes tag-based text templates for configuring said page." Reference 141 teaches tag-based text templates in Section 4.6.1.1, so claim 22 is anticipated.

507. Claim 22 is also anticipated by the Oracle 1.0 References. Claim 22 is identical to '335 claim 8 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '335 claim 8. Therefore, the Oracle 1.0 References teach all the limitations of claim 22.

508. Claim 23 adds the limitation to claim 22 "wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates."

509. Claim 23 is anticipated by reference 141 because dynamic insertion is taught: "The following place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1.

510. Claim 23 is anticipated by the Oracle 1.0 References. Claim 23 is identical to '335 claim 9 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '335 claim 9. Therefore, the Oracle 1.0 References teach the limitations of claim 23.

511. Claim 24 adds the limitation to claim 22 "wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request."

512. Claim 24 is anticipated by reference 141 because the purpose of the tag-based text templates is to drive the format of the data.

513. Claim 24 is anticipated by the Oracle 1.0 References. Claim 24 is identical to '335 claim 10 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '335 claim 10. Therefore, the Oracle 1.0 References teach the limitations of claim 24.

514. Claim 25 adds the limitation to claim 22 "wherein said tag-based text templates include HTML templates."

515. Claim 25 is anticipated by reference 141 because the tag-based text templates are HTML templates as taught in Sec. 4.6.1.2.

516. Claim 25 is anticipated by the Oracle 1.0 References. Claim 25 is identical to '335 claim 11 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '335 claim 11. Therefore, the Oracle 1.0 References teach the limitations of claim 25.

'335 Claim 26

517. Claim 26 is anticipated by reference 016, Gaffney, "Illustra's Web DataBlade Model," which was previously shown to anticipate '335 parent claim 15.

518. Claim 26 adds the limitation of "dynamically updating data at said one or more data sources." Illustra's Web DataBlade module is a comprehensive toolset for creating Web-enabled database applications that dynamically retrieve and update Illustra database content. p. 1, which meets the limitation.

519. Therefore the reference teaches dynamic updating of the data sources and thus anticipates claim 26.

520. Claim 26 is also anticipated by the Oracle 1.0 References. Claim 26 is identical to '335 claim 12 except that it refers to a different parent claim. The Oracle 1.0 References have already been shown to teach the limitation of '335 claim 12. Therefore, the Oracle 1.0 References teach the limitations of claim 26.

'335 Claim 27

521. Claims 27 and 28 are anticipated by any reference which anticipates '335 claims 13 and 14. This includes reference 173, Knudson, "Application Development with Microsoft's Internet Information Server," as shown in Exhibit 3.

'335 Claim 29

522. Claim 29 is anticipated by reference 161, Scholl et al. U.S. Patent 5,742,762. The reference teaches a system for real-time (dynamic) construction of web content by accessing one or more "managed networks" to obtain information, which is then inserted into web pages for return to the requesting client.

523. It is a computer-implemented method, so limitation 29a is met.

524. Limitation 29b is met because a request is transferred from an HTTP-compliant device (web server 3) to a dispatcher (network management gateway 5) (see Figs. 3, 4, 6), the steps being described at 6:6-31; 6:57-7:4. The routing is shown as an arrow from box 18 to box 11 in Fig. 3.

525. Limitation 29c is met because the network management proxy agents 15 keep information about the data sources available on the managed networks 6. This process is shown in Figs. 3, 4 and 6 and described at 7:5-35; 7:49-8:10. The Management Information Database (MIB) 14 of Fig. 4 maintains "a database of information regarding the managed networks 6 and information regarding the network management gateway 5." 7:2-5.

526. The page server function is performed by a combination of the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. The content of the web page is provided by the appropriate network management system, and HTML commands are added by the output formatter. Limitation 29d1 is met because the

network management gateway 5 makes an informed decision which managed network should process the request (dispatching). Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which “appropriate network proxy agent” should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen. Block 26 of Fig. 6 shows a test to determine whether the required information is available in a database. This process is described at 6:57-7:4. The Management Information Database (MIB) 14 of Fig. 4 maintains “a database of information regarding the managed networks 6 and information regarding the network management gateway 5.” (7:2-5)

527. Limitations 29d2 and 29d3 are met because the page server receives the request using TCP, which provides an acknowledgement of receipt: “The present invention 5 parses and translates the request, converts the request into the appropriate network management service requests, and forwards each request to the appropriate managed network 6 using the appropriate communication protocol (for example, TCP/IP, frame relay, CDPD, or others) for each managed network 6, through the appropriate Internet (or other) network access 7.” (6:15-21). This constitutes releasing according to epicRealm’s contention.

528. Step 29e is taught because it is analogous to step ’335 15d, which was shown above to be taught by the reference.

529. Likewise, steps 29f1 and 29f2 are analogous to steps ’335 15e1 and 15e2, which were shown above to be taught by the reference.

530. Therefore, claim 29 is anticipated by reference 161.

531. Claim 29 is also anticipated by the Oracle 1.0 References, which teach a computer-implemented method (preamble 29a). Steps 29b corresponds to step 16b, which has already been shown to be taught by the Oracle 1.0 References.

532. Page 4-5 of reference 153 (MAC003109) describes how the page server (Oracle7 Server) can access multiple databases and teaches a facility known as SQL*Net, which communicates with components of a distributed database. It is inherent in this structure that the

page server maintains information on which data sources it may access. Therefore, the Oracle 1.0 References teach limitation 29c.

533. Chapter 4 of reference 153 (beginning at page MAC003105) describes the use of P/SQL procedures to retrieve dynamic data. The information maintained in stored PL/SQL procedures specifies which Oracle7 Server (page server) is to be invoked to handle the request. The Web Agent dispatches the request to the appropriate page server based on the request (*e.g.*, its URL). Therefore, the Oracle 1.0 References teach the limitation 29d1 of "dispatching said request to an appropriate page server based on said request and based on said dynamic information."

534. Step 29d2 corresponds to '335 claim 15b2 and 15b3, which have already been shown to be taught by the Oracle 1.0 References.

535. Steps 29e1 and 29e2 are identical to '335 steps 15d1 and 15d2, which have already been shown to be taught by the Oracle 1.0 References.

536. Steps 29f1 and 29f2 are identical to '335 step 15e, which has already been shown to be taught by the Oracle 1.0 References.

537. Therefore, the Oracle 1.0 References teach all the limitations of claim 29 and anticipate it.

538. Claim 29 is anticipated by at least 29 references, as listed in Exhibit 4.

539. Numerous other references anticipate the claims, as listed in Exhibits 3 and 4. The basis for this opinion is given in the cells in the spreadsheet of Exhibit 3.

540. Oracle 2.0 (sometimes herein Oracle 2), described in the Oracle 2.0 References, incorporates all the functionality of Oracle 1.0 and more. Therefore, as shown in Exhibit 3, Reference 604/605 anticipates all of the claims that are anticipated by the Oracle 1.0 References. In addition, Reference 604/605 anticipates '554 claim 4: "To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection

cache. A later section of this report lays out precisely how The Oracle 2.0 References anticipate all claims of the Patents.

541. Reference 604/605 also teaches dispatching by load balancing: "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11. The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed as follows: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5.

ORACLE DEVELOPMENTS

542. In this report I have relied heavily on the Oracle 1.0 References and Oracle 2.0 References to demonstrate claim invalidity. No claim that reads on these references could possibly be valid since they are prior art. Nevertheless, it is useful to understand the historical development of the products described in the Oracle 1.0 References and Oracle 2.0 References to show how Oracle addressed the problem of dynamic web page generation.

543. In preparing this section I have relied on the Affidavit of Magnus Lönnroth dated March 26, 2007 ("Lönnroth Aff.") and the Deposition of Katrina Montinola ("Montinola Depo.") taken February 19, 2008.

544. As the World-Wide Web developed, a need arose to facilitate access to databases remotely over the Internet. Oracle was and is one of the world's largest suppliers of database systems, so to make its products more attractive Oracle software developers in 1994 began researching ways to integrate the Internet with Oracle's database products. For example, beginning in 1994, Oracle developers including Magnus Lönnroth and Joe Pistritto created Oracle software products for installation on web server computers which provided access to database information via the World Wide Web. One of those early products included the Web-Oracle-Web, or "WOW," Gateway. See Lönnroth Aff., ¶ 3 and Deposition of Montinola Dep. at 29:9-30:2.

545. Mr. Lönnroth wrote the source code for the WOW Gateway between September and December 1994, and Oracle released the WOW source code to the public for free on or before March 1, 1995. Lönnroth Aff. at ¶ 4. A copy of the WOW Gateway source code is attached to the Lönnroth Affidavit at Exhibit A.

546. As discussed in Lönnroth's Aff. at ¶ 6, the WOW Gateway was designed to connect a web server to an Oracle database server to provide access to database information suitably marked up with HTML for display on browsers over the World Wide Web. Once installed on a web server, the WOW Gateway provided this connectivity as follows. First, a request is received at the web server, which is configured so that requests for database information invoke a CGI program called "wowstub." When invoked, wowstub examines web page requests (including those for dynamic web pages) to determine which database application should process the request. Then, wowstub logs into the database server and passes the request to the appropriate application on that server which executes to query the database and dynamically generate an HTML web page. Finally, wowstub retrieves the resulting web page from the output buffer, and the web server returns the completed HTML document to the requester.

547. According to Mr. Lönnroth, upon its release on or before March 1, 1995, the WOW Gateway was put into use at various locations inside and outside the United States. Lönnroth Aff. at ¶¶ 7-8. After completing the code for the WOW Gateway, Joseph Pistritto and Mr. Lönnroth became involved in developing the Oracle WebServer, which was an enterprise-class package of software that was the commercial successor to the freely available WOW Gateway. Lönnroth Aff. at ¶ 9, Montinola Depo at 15:2-10, and ORCL01043070-091. Other members of the Oracle team that developed the Oracle WebServer included Katrina Montinola, Matthew Bookman, Brian Zuzga, Mala Anand, Kelly Ireland, and Aileen Jaitin. Montinola Depo at 15:11-21. The Oracle WebServer includes a Web Listener and a Web Agent and was designed to operate with the Oracle7 database server to provide access to database information in HTML format over the World Wide Web. This is described in the Oracle 1.0 References.

548. The Oracle WebServer together with the Oracle7 Database Server comprised a system for providing access to database information over the Web. See, generally, Oracle WebServer User's Guide (Release 1.0) at ORCL000002-224. Like the Oracle WOW Gateway, the Web Agent component of the Oracle WebServer provides the connectivity between a web server and a database server. Particularly, the Web Agent connects the Web Listener, a web server, to page generating applications residing on an Oracle7 database server. For example, a request is received at the Web Listener, which is a web server capable of handling requests for static or dynamic web pages. Requests for dynamic web pages are intercepted at the Web Listener by the Web Agent. The Web Agent examines such a request to determine which application on the database server should process the request. The Web Agent then logs into the Oracle7 database server and dispatches the request to the appropriate application. The appropriate application (i.e., the page server) executes to query the database and dynamically generate an HTML web page including the information sought by the request. This process is designed to operate asynchronously, so that while the application on the database server is processing a request, the Web Listener is free to process and respond to other requests. Lönnroth Aff. at ¶ 11.

549. The Oracle WebServer was released as a product in the fall of 1995. Lönnroth Aff. at ¶ 11, Montinola Depo. at 33:15-35:10, and ORCL000359-379. Particularly, the beta version of the Oracle WebServer was released for testing and reviewed by Oracle customers in September 1995, in connection with a demonstration and presentation made at the International Oracle User's Week trade show ("IOUW95") on September 25, 1995. Additionally, the Oracle WebServer was disclosed in detail in a technical paper presented and distributed by Katrina Montinola at IOUW95 on September 25, 1995. Lönnroth Aff. at ¶ 15, Montinola Depo. at 274:15-23, and ORCL01042968-974. The beta release of the Oracle WebServer was made available on CD to attendees at IOUW95 and/or hard copy to any other interested persons. Lönnroth Aff. at ¶ 15. According to Mr. Lönnroth, the beta release represented a finished product, and its architecture and operation were identical to the architecture and operation of the first commercial release of the Oracle WebServer 1.0 product. Lönnroth Aff. at ¶ 15. Additionally, according to Mr. Lönnroth and Ms. Montinola, Oracle made the WebServer 1.0 commercially available to the public no later than October 30, 1995. Lönnroth Aff. at ¶¶ 15-17 and Montinola Depo at 41:16-23.

550. Keith Lowery, one of the named inventors on the Patents, acknowledged in an email dated September 1, 1995 that "I have done research on compuserve on what Oracle is doing in this area since they are they main ones I have seen with an internet server. They are doing something similar but it is limited to surfacing HTML from and executed PL-SQL statement. They do not have the ability to build HTML templates for attractive formatting, and their solution is fundamentally proprietary to their own backend." EPIC000150. Mr. Lowery's email does not take into account that Oracle 1.0 indeed incorporated procedural HTML templates.

551. While Oracle was developing Oracle WebServer 1.0, it was concurrently developing WebServer 2.0. See, e.g., Montinola Depo. at 36:20-37:14 and 184:5-187:6, ORCL01043028-045, and ORCL01043092-101. For example, in a September 1995 conference paper, Katrina Montinola wrote: "In Release 2.0, which we have been developing in parallel with

Release 1.0, a key differentiating factor between the Oracle WebServer and traditional Web servers will be the ability to store Web objects, such as HTML documents and GIF files, or pointers to Web objects, in the database." Montinola Depo. at 274:15-23 and ORCL01042968-974. Oracle WebServer 2.0 operated similarly to WebServer 1.0 but also included some technical enhancements. For example, WebServer 2.0 included a Web Request Broker which dispatched dynamic web page generation requests to a variety of WRB Services including, for example, a Java Interpreter. See, generally, Oracle WebServer User's Guide at ORCL01816630-882, Design Specification for Oracle Web Agent and OWS Developer's Toolkit, Oracle Web Server, V2.0, Oracle WebServer Release 2.0 Feature List, Web Request Broker API Version 0.8., Web Request Broker API, and Oracle WebServer 2.0 Technical Note dated March 1996 at ORCL000761-781.

552. Oracle 2.0 contains numerous teachings not found in Bookman U.S. Patent 5,761,673, which described and claimed a portion of Oracle 1.0. In particular, as shown in Exhibit 3, Oracle 2.0 anticipates claims of the Patents not anticipated by Oracle 1.0. Furthermore, Oracle 2.0 uses several mechanisms not disclosed in Bookman, including: a different load balancing mechanism, a Web Request Broker, a Web Request Broker dispatcher, Web Request Broker Executable Engines, pools of processes for connection caching, page caching at the Web Listener, a LiveHTML Interpreter for including dynamic content in web pages, an "include" command for including dynamically generated content in web pages, a "FOR UPDATE" command to lock database tables for updating during a request, and methods for handling objects, including a Java Interpreter, CORBA and OLE interfaces,

553. Oracle released a beta version of WebServer 2.0 to the public in February 1996. Montinola Depo. at 38:9-39:6. Additionally, Oracle WebServer 2.0 was demonstrated to the public in February 1996 at the E-mail World & Internet Expo in San Jose, California. See ORCL01043388-390. Oracle then released a commercial version of WebServer 2.0 the following month in March 1996. Montinola Depo at 38:9-39:6, ORCL01043390, and ORCL000761-781.

554. Oracle WebServer 2.0 expanded the functionality of Oracle 1.0 by among others implementing various types of WRB Services (page servers) and implementing a WRB dispatcher to dispatch requests to those servers. All claims of the Patents are anticipated by Oracle 2 as follows.

555. As to the '554 patent, limitation of 1a is met because the documentation discloses that "using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." ORCL01816648. The "database is used for Web pages that are generated at runtime using 'live' data." ORCL01816643.

556. Limitation of 1b1 is met, because "[t]he HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL01043370. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL01043372.

557. Limitation 1b2 is met because "The Oracle WebServer comes with three WRB Services: the PL/SQL Agent, the Java Interpreter, and the LiveHTML Interpreter." ORCL01816808. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL01043377.

558. Limitation 1b3 is met because a diagram at page ORCL01606867 shows that a Web Application (such as PL/SQL Agent) sends a REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. Under epicRealm's releasing theory, with which I do not agree, HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more

efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Additionally, under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload the processing of the request to the now initialized WEB Application, prospectively freeing up the HTTP Server to process other requests once a request capable of being processed by the WEB Application is received.

559. Limitation 1b4 is met because "[t]he HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." ORCL01043772. This "handing off" without processing is intercepting.

560. Limitation 1b5 is met because "[t]he HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL01043770.

561. Limitation 1b6 is met because "[t]he WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." ORCL01043372.

562. Limitation 1c is met because "[w]hen the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." ORCL01043370. "Oracle's HTTP Server is

designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL." ORCL01043370

563. Limitation 1d1 is met because "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.

564. Limitation 1d2 is met by Oracle 2 because "The Oracle WebServer ... draws on information from the database and the operating system's (OS) file system as necessary to respond to the request. [T]he database is used for Web pages that are generated at runtime using 'live' data." ORCL01816643. Additionally, Oracle 2 meets this limitation as follows: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point." ORCL01043384-ORCL01043385.

565. The limitation of claim 2 is met because "The PL/SQL Agent uses Database Connection Descriptors (DCDs) to control the privileges an application runs under and the database schema to which it connects in a generalized and application independent way." ORCL01816683. "[PL/SQL Agent] also needs to have access to all the details required to actually establish a permanent connection: typically a set of environment variables (ORACLE_HOME and ORACLE_SID for UNIX systems) and perhaps a SQL*Net connect descriptor. All of this information is referred to as a DCD (Database Connection Descriptor), and all of the DCDs known t o the PL/SQL Agent are stored in a configuration file named owa.cfg (DCDs where known as Web Agent Services in Webserver 1.0). . . . DCDs are defined in the

PL/SQL Agent's configuration file, but they are activated by creating a directory mapping in the Web Request Broker's configuration file." ORCL01043378.

566. The limitation of claim 3 is met because "The Oracle WebServer ... draws on information from the database and the operating system's (OS) file system as necessary to respond to the request. [T]he database is used for Web pages that are generated at runtime using 'live' data." ORCL01816643. Additionally, Oracle 2 meets this limitation as follows: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point." ORCL01043384-ORCL01043385.

567. The limitation of claim 4 is met because "Due to the new architecture in Oracle Webserver 2.0 with the Web Request Broker, the PL/SQL Agent is roughly an order of magnitude faster than the previous version. This is mainly due t o the fact that each instance of the PL/SQL Agent stays connected to Oracle7 between requests, and thus does not need to establish a new database connection each time a stored procedure needs to be executed." ORCL01043377. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step. Since each HTTP request needs a new session, we are technically logging on and off between requests, but this is still extremely fast since the Oracle7 Server connection is maintained between requests." ORCL01043378. The limitation of claim 5 is met because "[a]t the lowest level, Oracle; internal API allows us to establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." ORCL01043378

568. The limitation of claim 6 is met because the documentation states, "File caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into

memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." ORCL01816646.

569. The limitation of claim 7 is met "LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." ORCL01816643. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages" ORCL01816656. "LiveHTML code is formatted as SGML comments ... The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." ORCL01816697. "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." ORCL01816803.

570. The limitation of claim 8 is met because ". . . include. This command specifies that a file is to be included in the generated HTML page at this point. The file can any of the following: - another LiveHTML file like the current one. - a regular HTML file. - an ASCII file." p. 4-30

571. Limitation 9a because "using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." ORCL01816648.

572. Limitation 9b is met as follows: "The Oracle WebServer ... draws on information from the database and the operating system's (OS) file system as necessary to respond to the request. [T]he database is used for Web pages that are generated at runtime using 'live' data." ORCL01816643. Additionally, Oracle 2 meets this limitation as follows: "LiveHTML code is

formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1=”value1” tag2=”value2” --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point.” ORCL01043384-ORCL01043385.

573. Limitation 9c is met because "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension.” ORCL10143370. “The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues.” ORCL01043372. Thus the WRB services are “page servers” of the patent. A processor of a computer that runs WRB services is the page server processing means.

574. Limitation 9d is met because the first computer system includes the user’s web browser. "From a web browser's point of view, Oracle WebServer 2.0 looks just like any other web server." Technical Note, p. 2. Also, the first computer is a computer that has a processor that runs a web browser as shown on page 1-3. See ORCL000763. Therefore, means for generating said request is a processor of a computer running a web browser as show in the figure at ORCL000763.

575. Limitation 9e1 as met because the second computer system includes the Web listener, the Web Request Broker, as shown on ORCL01043369, and the WRB Executable Engines (page servers).

576. Limitation 9e2 is met because "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL01043370. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL01043372. Means for receiving said request is the processor of a computer that runs the Web Listener. See the figure on ORCL01043369.

577. Limitation 9e3 is met by the module code that performs the routing at the second computer system is the router.

578. Limitation 9e4 is met because "The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." ORCL01043372. The "handing off" without processing is intercepting.

579. Limitation 9e5 is met because "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL01043370.

580. Limitation 9e6 is met as because the WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." ORCL01043372. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the

WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." ORCL01043372.

581. Limitation 9e7 is met because "[t]he PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL01043377.

582. Limitation 9e8 is met because the Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Additionally, under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload the processing of the request to the now initialized WEB Application, prospectively freeing up the HTTP Server to process other requests once a request capable of being processed by the WEB Application is received.

583. Limitation 9e9 is met because "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure by the processor of the computer running PL/SQL Agent brings up a page that looks like page shown at ORCL01816677. Also see ORCL01816678-79.

584. Limitation 9e10 is met because "The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the

request. The file system can be used for static (hardcoded) Web pages, or CGI scripts that do not access the database, and the database is used for Web pages that are generated at runtime using 'live' data." ORCL01816643. Additionally, Oracle 2 meets this limitation as follows: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL01043369.

585. Claim 10 depends from claim 9. Limitation 10a is met by Oracle 2 because claim 9 is anticipated by Oracle 2.

586. Limitation 10b is met because "[t]he HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." ORCL01043372. This "handing off" without processing is intercepting.

587. Limitation 10c is met because the [t]he WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." ORCL01816808. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." ORCL01043372.

588. ’554 claim 11 is drawn to machine-readable medium. Oracle 2 software resides on machine readable media.

589. The functions 11b1-11b6 correspond to process steps 1b1-1b6. Therefore these limitations are met by Oracle 2.

590. Functions 11c and 11d correspond to process steps 1c and 1d1, 1d2 together. Therefore these limitations are met by Oracle 2.

591. Therefore, all claims of the ’554 Patent are anticipated by Oracle 2.

592. As to ’335 claim 1, steps 1a, 1b1-1b3, 1c1-1c2, 1d, 1e1-2 are analogous to ’554 claim 11 steps 11a, 11b1-1b4, 11b6 and 11d, all of which were shown to be met by Oracle 2.

593. Limitation 2a of claim 2 is met because routing occurs in Oracle 2 as discussed above in connection with limitation 1b1 of ’554. Limitations 2b-2c of claim 2 correspond to steps 1b5 and 1b6 of ’544 claim 1.

594. The limitations of claims 3-7 correspond to the limitations of ’554 claims 2-6. Therefore, ’335 claims 3-7 are anticipated by Oracle 2.

595. The limitation of claim 8 refers to “tag-based text templates” and it is met by Oracle 2 as follows: "LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." ORCL01816643. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages" ORCL01816656. "LiveHTML code is formatted as SGML comments ... The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." ORCL01816697 "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." ORCL01816803.

596. The limitation of claim 9 is met because "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." ORCL01816803.

597. The limitation of claim 10 is met because: "LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." ORCL01816643 "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages" ORCL01816656. "LiveHTML code is formatted as SGML comments ... The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." ORCL01816697. "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." ORCL01816803.

598. The limitation of claim 11 is met because: "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." ORCL01816803.

599. The limitation of claim 12 is met as follows: "You can also use a cursor to update a table's contents. To do this, use a FOR UPDATE clause to lock the rows in the table. See "Using FOR UPDATE" in the PL/SQL User's Guide and Reference for more information. Sometimes, you may need to use cursor variables. which are not associated with a query until runtime. This is a form of dynamic SQL." ORCL01816828.

600. The limitation of claim 13 and 14 are met because "[t]he Java Interpreter includes classes for native Oracle7 access via PL/SQL and dynamic creation of HTML documents. For customers who wish to employ their own programming language o f choice, Oracle WebServer 2.0 offers an easy-to-use API allowing you to build your own server-side plugins ranging from existing legacy applications written COBOL to modern CORBA or COM/OLE based object-oriented systems written in for instance C++ or Smalltalk." ORCL01043368-ORCL01043369.

601. Limitation 15a is met because Oracle 2 performs a computer implemented method.

602. Limitation 15b1, "transferring a request from an HTTP-compliant device to a page server," is met by Oracle 2 because the Web Listener is HTTP-compliant and "transferring" and "routing" have the same meaning. Since Limitation 1b1 of '554 is met by Oracle 2, so is limitation 15b1 of '335.

603. Limitation 15b2 is identical to limitation 1b2 of '554, hence is met by Oracle 2.

604. Limitation 15b3, "releasing said HTTP-compliant device to process other requests" is a general case of limitation 1b3 of '554, hence is met by Oracle 2.

605. Limitation 15c, "intercepting said request at said HTTP-compliant device and transferring said request to said page server" is a general case of limitation '335 1c1 and 1c2, hence is met by Oracle 2.

606. Limitations 15d1 and 15d2 together are a general case of '335 limitation 1d, hence are met by Oracle 2.

607. Limitation 15e is essentially identical to limitations 1d1 and 1d2 of '554, hence is met by Oracle 2.

608. Limitation 16a is perforce met by Oracle 2.

609. Limitation 16b is a general case of limitation 1b5 of '554, hence is met by Oracle 2.

610. Limitation 16c is identical to limitation 1b6 of '554, hence is met by Oracle 2.

611. Therefore, '335 claim 16 is anticipated by Oracle 2.

612. The limitations of claims 17-21 are essentially identical to the limitations of '554 claims 2-6, respectively, hence are met by Oracle 2.

613. The limitations of claims 22-28 are essentially identical to the limitations of '335 claims 8-14, respectively, hence are met by Oracle 2.

614. Limitation 29a is met by Oracle 2 because it is a computer-implemented method.

615. Limitation 29b is essentially identical to limitation 16b of '335, hence is met by Oracle 2.

616. Limitation 29c is met by Oracle 2 "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues. The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service t o have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." ORCL01043372. "The Dispatcher continually adjusts the load by controlling how many WRBXs are running at a given time, subject to certain parameters. These parameters are set by the Webserver Administrator, who sets the maximum and minimum number of WRBXs running for each WRB Service. The Dispatcher creates new WRBXs as needed and connects them to the appropriate WRB Services. The Dispatcher also keeps track of which WRBXs are executing requests and which are free." ORC01816653.

617. Limitation 29d1 requires "dispatching said request to an appropriate page server based on said request and based on said dynamic information." Oracle 2 meets this limitation because The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created

and destroyed according to workload." ORCL01816808. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5 (ORCL000766).

618. Limitation 29d2 is a general case (HTTP-compliant device) of limitations 9e7 and 9e8 of '554, hence is met by Oracle 2.

619. Limitations 29e1 and 29e2 together are a general case (HTTP-compliant device) of limitations 1b6 and 1c of '554, hence are met by Oracle 2.

620. Limitation 29f1 is identical to limitation 1d1 of '554, hence is met by Oracle 2.

621. Limitation 29f2 is essentially identical to limitation 1d2 of '554, hence is met by Oracle 2.

622. Therefore, all claims of the '335 Patent are anticipated by Oracle 2.

ALL OF THE ASSERTED CLAIMS ARE OBVIOUS

623. The asserted claims of the '554 and '335 Patents merely claim well known, commercially available machines and techniques for performing the functions enumerated in the claims, including connection caching, custom HTML extension templates, dispatching, driving a format of the data, dynamic Web page generation, dynamically retrieving data, dynamic updating, HTML templates, an HTTP-compliant device, intercepting, logging into a database, object handling extensions, OLE extensions, page caching, releasing, routing and tag-based text templates. Each of these is used in the Patents exactly as it was intended to be used and was used in the prior art. There is no new concept or construct that was invented by Applicants.

624. During 1994 and 1995, the need for fast dynamic page generation was widely recognized throughout the industry, and numerous systems were in parallel development by multiple companies to offload time-consuming dynamic page requests to separate servers.

625. I am informed that the common sense of those skilled in the art can demonstrate why combinations of prior art references would be obvious, that if adding a well-known prior art element is not uniquely challenging or difficult then such an addition can be obvious and that if one of ordinary skill in the art would have an apparent reason to combine known elements then the combination is obvious, and showing the use of common sense after the fact is not engaging in impermissible hindsight. One of ordinary skill in the art is able to fit the teaching of multiple references together like pieces of a puzzle if common sense suggests using known solutions to the problem at hand, and such a combination would then be obvious.

626. The problem of handling dynamic page generation requests efficient was not only well-known in the field, as evidence by the large number of anticipating prior art references discussed above, but numerous solutions, including each and every one of the solutions claimed in the Patents, had already been implemented in the prior art.

627. Thus the ’554 and ’335 Patents merely claim predictable uses of old elements according to their established functions. Combining these old elements was well within the level of ordinary skill in the art to implement, and, in each claimed combination, each element merely performs the same function it had previously been known to perform. No new or different functions are performed by the elements in combination, none are recited in the claims, and the arrangements all yield no more than one of ordinary skill would have expected from such arrangements, and no new fruitful results were achieved.

628. epicRealm cannot point to any unexpected advantage or result flowing from anything novel disclosed and claimed in the ’554 and ’335 Patents, and not provided by the prior art. I also understand that adding routine details well-known in the prior art to another item or prior art does not entitle the claimed subject matter to patent protection.

629. A claim is obvious if all of its limitations and those of its parent claims are found in a combination of references (that is, two or more rows of the spreadsheet of Exhibit 3) and if there is reason to combine the references. Every one of the references of Exhibit 3 deals with

dynamic web page generation by means of page servers. Therefore they are all in the same narrow specific art, which provides the motivation to combine.

Single-Reference Obviousness

630. I understand that a claim can be obvious in light of a single reference, without the need to combine references, if the elements of the claim that are not found in the reference can be supplied by the common sense of one of skill in the art.

631. To the extent that the following references may not be found to anticipate the claims listed as anticipating in this report (including the Exhibits), those claims are obvious in light of each reference individually.

632. The following steps would be supplied by the common sense of one of skill in the art of dynamic web page generation in August 1995. This is true in light of the extensive "Prior Art" section of this Report, which explains the development of various features of web-based systems.

633. It would be common sense to apply the well-know three-tier and four-tier architectures to achieve efficiency. A shared feature of both of these architectures is the off-loading of processor-intensive work from the web server tier (Tier 2) to a subsequent tier. If the web server were capable of handling all requests, there would be no need for additional tiers. In dynamic page generation, the processor-intensive work is page generation, so the Tier 3 server to which dynamic requests would go can be called a "page server."

634. In a multi-tier architecture, requests must be routed from Tier 2 to Tier 3 to be processed at Tier 3. This is step 1b1 of '554 claim 1.

635. The server at Tier 3 must receive the request in order to process it. This is step 1b2 of '554 claim 1.

636. The entire purpose of the multi-tier architecture is to free the Tier 2 (web) servers to process other requests. Were they not freed, there would be no point in off-loading requests to Tier 3. According to epicRealm's contentions, mechanisms of sending messages or establishing connections, such as socket()-bind()-listen() and TCP, free resources of the sender hence free it

to process other requests. TCP and socket()-bind()-listen() were widely used in web architectures, so it would be common sense to use them for routing requests from Tier 2 to Tier 3. This is step 1b3 of '554 claim 1.

637. Intercepting is a necessary step in multi-tier architectures. Some evaluation of a request must be made to intercept the handling of a request a Tier 2 to prevent a web server from handling it. Otherwise, the request would never be sent to Tier 3. This is step 1b4 of '554 claim 1.

638. It would be common sense to utilize more than one page server at the Tier 3 level to achieve scalability. It was well-known in multi-tier systems to use multiple servers at Tier 3, as shown in the "Prior Art" section of this Report.

639. If multiple page servers are employed at Tier 3, some decision is necessary to decide which server should receive a request. The threshold decision is which subset of page servers are capable of handling the request (i.e. which ones have access to the necessary data sources), which involves examining the request. Several ways of deciding which Tier 3 server to use were well known in the art, thus all of which would have been within the common sense of those skilled in the art. One of the well-known methods was load-balancing, which was known decades before the Patents were filed. It would be common sense to distribute requests to multiple page servers by load balancing. The load balancer is the dispatcher of step 1b5 of '554 claim 1.

640. Load balancing requires maintaining dynamic information about page servers (i.e. their instantaneous loads). Load balancing dispatching is therefore step 1b6 of '554 claim 1.

641. Because the Tier 2 (web) server is released when the request is routed to Tier 2, the Tier 2 server is able to process additional requests concurrently while the Tier 3 server generates a dynamic page. This is the point of the multi-tier architecture and is step 1c of '554 claim 1.

642. Because the stated problem is to handle a dynamic page generation request, one must generate a dynamic web page, which is step 1d1 of '554 claim 1. This does not even require common sense, but is part and parcel of the problem.

643. What separates a dynamic page from a static one is the fact that it is generated on the fly. Unless the page requires dynamic information, there is no point to generating it on request. Otherwise, it could have been kept as a static page. Therefore, common sense dictates that a dynamic page will contain data retrieved from at least one data source. This is step 1d2 of '554 claim 1.

644. This demonstrates that the nature of the problem itself, namely the handling of multiple dynamic web page generation requests, suggests using the well-known multi-tier architecture. The nature of the multi-tier architecture suggests all of the steps of '554 claim 1.

645. In the event that multiple data sources are available at Tier 3, it is necessary to identify which one(s) should be used to supply data for the request. Without such an identification, the request cannot be processed. This is the step of '554 claim 2. The use of multiple data sources in multi-tier architectures was well known in the art, and gave rise to such database connectors as ODBC.

646. If a page requires data dynamically retrieved from a data source, then dynamically retrieving that data is inherent. This is the step of '554 claim 3.

647. It was well known to maintain open connections to data sources to avoid the overhead of establishing and tearing down connections. One of the features of the problem as stated in the Patents is to handle dynamic page generation requests efficiently. One of the common sense methods of doing so is to maintain a cache of connections, as taught by numerous prior art references, including the Connection Cache References. This is the step of '554 claim 4. The fact that there are 23 Connection Cache References listed above indicates that the technique was so widespread as to be suggested by common sense to one of skill in the art.

648. It was well known to require login credentials to access data sources because of the generally proprietary nature of business data. If access to such a data source is necessary to generate a dynamic page, it is inherent that logging in would be required. It is also taught, for example, by the Logging In References. This is the step of '554 claim 5. The fact that there are

22 Logging In References listed above indicates that the technique was so widespread as to be suggested by common sense to one of skill in the art.

649. It was well known to store dynamic pages for delivery in response to subsequent requests to avoid the overhead of generating them anew each time they were requested. This storage was known as a page cache and is taught in numerous prior art references, including the Page Cache References. If an objective is to handle dynamic page generation requests efficiently, common sense would suggest using a page cache to improve efficiency. This is the step of '554 claim 6. The fact that there are 17 Page Cache References listed above indicates that the technique was so widespread as to be suggested by common sense to one of skill in the art.

650. Claim 9 of the '554 Patent attempts to claim an apparatus for carrying out the steps of '554 claim 1, other than intercepting. There is nothing novel in the apparatus, for each element performs the function expected of it. Therefore, to the extent that common sense suggests the steps of claim 1 to solve the problem of efficient handling of dynamic page generation requests, common sense also dictates the three-tier architecture of '554 claim 9. Element 9e3 claims a "router," but this element simply implements the routing step of '554 1b1.

651. Claim 10 of the '554 Patent attempts to claim an apparatus for carrying out the steps of '554 claim 1, including intercepting. For the same reason as stated for claim 9, there is nothing novel in the apparatus, which is simply a common sense system implementation of claim 1 using a multi-tier architecture.

652. Claim 11 of the '554 Patent claims a machine-readable medium containing computer instruction for carrying out the steps of '554 claim 1. There is nothing disclosed or claimed about the medium other than the steps performed by the code, all of which were shown above to be suggested by common sense applied to the problem of efficient handling of dynamic page generation requests.

653. All of the steps of ’335 claims 1-6 are steps that appear in ’554 claims 1-11. Therefore, as shown above, all are suggested by common sense applied to the problem of efficient handling of dynamic page generation requests.

654. Generation of dynamic web pages on a large scale requires the use of templates. Otherwise, a different piece of code or script would be required for every possible different dynamic page. It was well known to generate dynamic page by inserting dynamically retried data into tag-based text templates. This is taught for example, by the Tag Template References. This is the step of ’335 claim 8. The fact that there are 43 such references listed above indicates that the technique was so widespread as to be suggested by common sense to one of skill in the art.

655. The nature of templates is that they contain placeholders for data to be filled in. The step of ’335 claim 9, namely inserting dynamically retrieved data into the template, is therefore necessary. (If there were no dynamic data, no template would be required and the page could be maintained statically.) Therefore the step of ’335 claim 8 necessarily implies the step of ’335 claim 9. To the extent that claim 8 is dictated by common sense, so is claim 9.

656. The purpose of a template for generating a dynamic web page is to determine the format of the page. The template indicates which dynamic data is to be retrieved, where it should be placed, and how it relates to fixed textual portions of the page. Therefore, the templates necessarily drives a format of the data dynamically retrieved, which is the step of ’335 claim 10. Therefore, to the extent that common sense suggests the use of templates in handling dynamic page generation requests, the step of claim 10 necessarily follows.

657. Generation of dynamic pages requires access to dynamic data. It was well known to use database management systems, such as SQL server, to access dynamic data. SQL dates from the 1970s, and was familiar to those skilled in the art. One of the fundamental commands in SQL is UPDATE, which allows updating of a data source during a query. Therefore, updating is inherent in the use of SQL. Since common sense would suggest the use of SQL-compliant

database retrieval mechanisms, the ’335 claim 11 step of dynamically updating data at data sources is a consequence of data retrieval through SQL.

658. Claim 15-24 and 26 of the ’335 patent claim steps that appear in ’554 claims 1-11 and ’335 claims 1-12, all of which were shown above to be suggested by common sense applied to the problem.

659. Common sense of one of skill in the art would suggest that if tag-based text templates were used in dynamic web page generation, then the template should be HTML templates. The simple reason is that HTML is the language of web browsers and “web page” has been construed to mean “Web content displayable through a Web browser.” Therefore the very nature of dynamic web page generation suggests the use of HTML. If HTML needs to be part of the final generated page, then common sense dictates placing HTML in the corresponding template.

660. The foregoing demonstrates that ’554 claims 1-6 and 9-11 and ’335 claims 1-12, 15-26 and 29 are suggested by the common sense of one skilled in the art at the time the claimed inventions were allegedly made.

661. The remaining claims, ’554 claims 7-8 and ’335 claims 13-14 and 27-28, which are all essentially three pairs of the same claims, were not suggested by common sense but are obvious in light of combinations of references. Microsoft Internet Information Server (IIS), described in Reference 501/502, was widely used for dynamic generation of web pages. References 172 and 173 describe applications of IIS in which the object handling extension OLE is used to process dynamic page generation requests. The combination is suggested by the fact that all the references pertain to the same system. Therefore these claims are obvious.

662. All the claims of the Patents except ’335 claims 13, 14, 27 and 28 (the “OLE Claims”) are obvious in light of each of these single references alone: Varela et al. Reference 002, Clausnitzer et al. Reference 063, Derler Reference 141, Scholl et al. Reference 161, Popp Reference 194 and Oracle 1.0 Reference 153.

663. As a general matter, to the extent any element of any non-OLE Claim is not expressly taught in these references, one of ordinary skill in the art, facing the wide range of needs created by developments in the field, would have seen an obvious benefit to the solutions tried by the applicant. The needs created by developments in the field are discussed in detail in above. They are the need for efficiency and scalability in handling dynamic page generation requests, intercepting to take advantage of multi-tier architectures, and dispatching to improve efficiency in passing requests from Tier 2 to Tier 3.

664. Generation of dynamic web pages requires some form of skeleton, or template for the pages. Without a template, there would be no practical way to insert dynamically retrieved data into the page. Because web pages were typically composed using HTML at the time of the Patents, common sense suggests the use of HTML templates.

665. HTML itself does not provide a mechanism for accessing the data needed to generate dynamic pages. Adding a database call to an HTML file requires some extension to HTML. Many such extensions were known. If templates are used to specify the structure of web pages, then common sense suggests using HTML extension templates to specify the structure of dynamic pages, since some method must be provided to express calling for dynamic data. The purpose of having the template specify a data source is so that data returned can be inserted into the template.

666. Retrieving information dynamically is done by querying a database system. The most common method of expressing database queries at the time of the Patents (and still today) is through SQL. One of the SQL primitives commands is UPDATE, so common sense suggests updating data sources as part of a query. In fact it was common to do so. Even static web pages often contained a "hit counter," which would be incremented by one each time the page was delivered. This way the viewer could see how popular the page was. This is dynamic updating in a simple form.

667. Considerations of efficiency would call to mind to one of skill in the art to use caching to reduce the time needed to generate dynamic pages. Both connection caching to

reduce connection time and page caching to reduce the need to generate a page at all were well-known, obvious efficiency techniques.

668. The need to control access to private or corporate data calls to mind logging in to data sources, which long predated the Patents and was a virtually ubiquitous technique to prevent unauthorized persons from obtaining dynamic pages with proprietary information.

669. Even if the prior art does not anticipate the claimed inventions, e.g. because of the dates of such references, there is ample evidence of contemporaneous development of the claimed inventions by others, which is a factor to be assessed in determining obviousness. The large number of prior, near-simultaneous or simultaneous developments listed in Exhibit 3 (at least 30) which predate or are contemporaneous with the Patents show that numerous companies and researchers independently arrived at the solutions claimed in the Patents at about the same time.

670. At the Webmania 1996 conference, NeXT Software chairman Steve Jobs gave a public talk on NeXT's WebObjects product. He explained that WebObjects created dynamic web pages using data retrieved from data sources and returned the web pages to a web server to be transmitted to the requestor over the Internet. He illustrated a three-tier architecture for dynamic web page generation and predicted that dynamic content would be "Act 2" of the Web. He stated that WebObjects existed as of the date of his talk and had already been downloaded 1000 times.

671. I have reviewed the Cold Fusion 1.0 User Guide and the transcript from the recent deposition of Joseph J. Allaire, the creator of the Cold Fusion product. Mr. Allaire's deposition further illuminates the knowledge and common sense of those skilled in the art as of the time the claimed inventions were allegedly made. Specifically, Mr. Allaire explained that Cold Fusion 1.0 could have been implemented among a cluster of web server machines, and that a person of ordinary skill would have known to load balance requests from a first web server to multiple other web servers running Cold Fusion. In addition, Mr. Allaire explained how the Cold Fusion product documentation, combined with the knowledge of one ordinary skill, discloses techniques

for performing the functions enumerated in the claims, including connection caching, dispatching, driving a format of the data, dynamic web page generation, dynamically retrieving data, dynamic updating, HTML templates, logging into a database, releasing, routing, and tag-based test templates.

Multiple-Reference Obviousness

672. To the extent that any claim of the Patents may not be found anticipated or obvious in light of a single reference, it is possible to generate a huge number of additional obviousness combinations by examining the rows of the spreadsheet in Exhibit 3. Text in a spreadsheet cell means that the element in that column is found in the reference in that row. If the rows are considered in pairs, then a new hypothetical row can be constructed, corresponding to the combination of the two references, which will have text wherever either of the two rows has text. Such a combination would be obvious because all of the references related to the same art, that is, the art of generating dynamic web pages. In is my opinion that all such two-row combinations (of which there are almost 400), are obvious. In the paragraphs below I explain exactly why one of skill in the art would be motivated to make such two-reference combinations.

673. A similar exercise is possible using Exhibit 5, which includes far more references than the 20 mapped out in Exhibit 3. Exhibit 5 can be used to generate tens of thousands of two-reference obviousness combinations.

674. While multiple references have been shown above to anticipate and/or render obvious each of the Asserted Claims, there are yet other references that can be applied, as shown in Exhibit 3. Some examples, which do not constitute a complete list, follow.

675. All of the Asserted Claims are anticipated or rendered obvious by numerous combinations of just two references. All of the combinations below are motivated because all the references are from the art of dynamic web page generation.

676. Claims 12 and 26 of the ’335 Patent are obvious in light of any of references 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 018, 058, 066, 141, 161 or 196 combined with any one of the Dynamic Updating References 016, 019, 039, 051, 153, 162, 172, 173, 180, 194, 195, 401,

501/502, 601, 604/605 or 607. The motivation to combine is that at least the following references each expressly teach dynamic updating as part of the process of dynamic web page generation: 016, 153, 162, 163, 180, 194, 195, 401, 501/502 and 604/605, as shown in Exhibit 3.

677. Claims 13, 14, 27 and 28 of the ’335 Patent are obvious in light of any of references 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 039, 051, 058, 066, 141, 153, 161, 162, 180, 194, 195, 196, 501/502, 601, 604/605 or 607 and the OLE References 172, 173 or 401. The motivation to combine is that at least references 172, 173 and 401 expressly teaches the use of OLE as part of the process of dynamic web page generation, as shown in Exhibit 3. OLE is an “object handling extension,” so the teaching of OLE renders claims 13 and 27 obvious.

678. Claim 4 of the ’554 Patent and claims 5 and 19 of the ’335 Patent are obvious in light of any of the references which anticipate ’554 claim 1, namely 002, 00H, 00J, 00K, 010, 018, 058, 063, 173, 194, 196, 401, 601, or 607 combined with any one of the Connection Cache References 002, 00O, 00Q, 00W, 016, 019, 051, 141, 153, 161, 162, 172, 180, 195, 501/502 or 604/605. The motivation to combine is that at least the following references each expressly teach a connection cache in processing dynamic web page generation requests: 002, 00W, 016, 141, 195 and 604/605, as shown in Exhibit 3.

679. Claim 5 of the ’554 Patent and claims 6 and 20 of the ’335 Patent are obvious in light of any of the references which anticipate ’554 claim 1, namely 00H, 00J, 00O, 00Q, 00W, 018, 039, 051, 058, 161, 162, 194, 401 or 601 and any one of the Logging In References 002, 00K, 010, 016, 019, 141, 153, 172, 173, 180, 195, 196, 501/502, 604/605 and 607. The motivation to combine is that at least the following references each expressly teach logging in to a data source to process dynamic web page generation requests: 00K, 016, 058, 141, 153, 152, 169, 173, 180, 196, 501/502, 601 and 604/605, as shown in Exhibit 3.

680. Claim 6 of the ’554 Patent and claims 7 and 21 of the ’335 Patent are obvious in light of any of the references which anticipate ’554 claim 1, namely 00J, 00K, 00O, 00Q, 00W, 010, 018, 019, 058, 162, 172, 173, 180, 194, 195, 601 or 607and any one of the Page Cache References 002, 00H, 016, 051, 066, 141, 153, 161, 196, 501/502, 601 or 604/605. The motivation to combine is that at least the following references each expressly teach caching dynamic pages in processing dynamic web page generation requests: 002, 016, 066, 141, 153 and 604/605, as shown in Exhibit 3.

681. Claims 7 and 8 of the ’554 Patent are obvious in light of any of the references which anticipate ’554 claim 1, namely 00J, 00O, 010, 018, 051, 161, 162 or 180 and any one of the HTML Extension References 002, 00H, 00K, 00Q, 00W, 016, 019, 039, 058, 141, 153, 172, 173, 194, 195, 196, 401, 501/502, 601, 604/605 and 607. The motivation to combine is that at least the following references each expressly teach HTML extension templates for processing dynamic web page generation requests: 002, 00K, 00Q, 00W, 016, 019, 052, 058, 061, 141, 153, 173, 194, 195, 501/502 and 604/605, as shown in Exhibit 3.

682. Claims 10 and 11 of the ’554 Patent and claim 29 of the ’335 Patent are obvious in light of reference 039 and the Dispatching References 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 019, 051, 058, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 or 607 (all of which also teach intercepting). The motivation to combine is that at least the following references each expressly teach dispatching in handling dynamic web page generation requests: 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 019, 051, 058, 141, 153, 161, 162, 172, 173, 180, 194, 195, 196, 401, 501/502, 601, 604/605 and 607, as shown in Exhibit 3.

683. Claims 8, 9, 22 and 23 of the’335 Patent are obvious in light of any reference which anticipates ’335 claim 1 but lacks tag-based text templates, namely 010 or 161 and any

one of the Tag Template References 002, 00J, 00K, 00Q, 00W, 016, 018, 058, 063, 141, 153, 173, 180, 194, 195, 501/502 or 604/605. The motivation to combine is that at least the following references each expressly the use of tag-based text templates in generating dynamic web pages: 002, 00J, 00K, 00Q, 00W, 016, 058, 141, 173, 180, 194, 195, 501/502 and 604/605, as shown in Exhibit 3.

684. Claims 10, 11, 24 and 25 of the ’335 Patent are obvious at least in light of any reference which anticipates ’335 claim 1 but lacks HTML extension templates, namely 00O, 010, 066 or 161 and any one of the HTML Extension References 002, 00H, 00K, 00Q, 00W, 016, 019, 039, 058, 141, 153, 172, 173, 194, 195, 196, 401, 501/502, 601, 604/605 and 607. The motivation to combine is that at least the following references each expressly teach driving a format of the data by HTML templates in generating a dynamic web page: 002, 00K, 00Q, 00W, 016, 052, 058, 141, 153, 173, 194, 195, 501/502 and 604/605, as shown in Exhibit 3.

685. Claims 14 and 28 of the ’335 Patent are obvious in light of any reference which anticipates ’335 claim 1 but lacks OLE, namely 002, 00H, 00J, 00K, 00O, 00Q, 00W, 010, 016, 018, 019, 039, 051, 058, 066, 141, 153, 161, 162, 180, 194, 195, 196, 501/502, 601, or 604/605 and the OLE References 172, 173 and 401. The motivation to combine is that at least references 172, 173 and 401 expressly teache the use of OLE as part of the process of dynamic web page generation, as shown in Exhibit 3. OLE is an “object handling extension,” so the teaching of OLE also renders claims 13 and 27 obvious.

686. Garland reference 013 teaches that it would be obvious to combine its distributed Web server architecture and load balancing techniques with any system (e.g., a two-tier or three-tier web server architecture) that suffers from the drawbacks identified in the Introduction, namely, the problem of having a single server receiving and processing too many requests. Garland teaches that requests can be divided up and dispatched among two or more distributed

and parallel servers such that server load can be spread out among many machines that have independent processing power and such that the front-end server (e.g., a web server or dispatcher server) and the several member servers (e.g., page servers) can concurrently process requests. See, generally, Garland at Section 3.

687. Garland reference 013 also introduces a motivation to combine historic load balancing techniques with the management of web page requests in a web architecture such as that disclosed in Garland's Figure 5. Particularly, Section 8 titled "Bibliography" cites to a variety of load balancing prior art references that describe load balancing techniques in various system and server architectures that are not necessarily in the field of Internet technology. Garland, therefore, demonstrates a motivation to combine traditional system load balancing techniques with more modern server architectures that are used for the management of web page requests including dynamic web page requests.

688. To the extent that TCP releasing and/or CORBA releasing and/or SBL releasing are not found to be "releasing" within the meaning of the claims, express releasing is taught by L. Stein in the book *How to Set Up and Maintain a World Wide Web Site*, published in 1995 (Bates H-COM-P-ART 916-933). Stein's page server is a background process which generates a response to a request received by an HTTP server (Web server). Stein teaches a method of ensuring that the Web server does not remain tied up while the background process satisfies the original request. In particular, Stein discloses that "[t]he main tricky point when launching a background process from a CGI script is that the child process must close standard input, standard output and standard error in order to truly go into the background. Otherwise its connection with the HTTP server will remain open. This is why the first thing the child process does after forking is to close the filehandles STDOUT, STDERR and STDIN." (H-COM-P-ART 000924). Because Stein deals with Web sites and the generation of pages for websites, and all of

the references in Exhibit 3 pertain to generation of web pages, it would be obvious to combine Stein with such references to render releasing obvious.

689. A particularly easy obviousness demonstration is possible with reference 194, Popp et al. U.S. Patent 6,651,108 combined with the NeXT Computer Inc. reference 354 (Exhibit 3) and, as required, with reference 195, Popp et al. U.S. Patent 6,249,291 (reference 195). The Popp et al. patents are assigned to NeXT Computer, Inc. and describes the same system referred to in the references. The motivation to combine, therefore, derives from the fact that all the references describe the same system.

690. As to the '554 Patent, Popp teaches a method of generating object-oriented web pages in response to requests to an HTTP (web) server 206. Claim 9 is drawn to dynamically generating a web page. Limitation 1a of '554 is therefore met.

691. The request is routed (sent) from the HTTP server 206 to the software application 214 (Fig. 2) which generates the dynamic page. Application 214 fits the Texas Court's construction of "page server" in that it is "page-generating software that generates a dynamic Web page." Limitation 1b1 is therefore met.

692. The request is transmitted to application 214 at 7:5-8. Because the application acts on the request (7:8-9), it must receive it. Limitation 1b2 is therefore met.

693. Popp et al. does not expressly teach limitation 1b3, releasing the web server. However, such releasing is described on p. 2 of reference 350. Limitation 1b3 is therefore met by the combination.

694. Limitation 1b4, intercepting the request at the web server, is shown in Fig. 2 and described at 6:61-65; 7:5-9.

695. The "dispatcher" of the '554 Patent is the "CGI Messenger" 210. The required "informed decision" as to which page server (application) should be invoked is based on the "app_name," as described at 6:52-59. Limitation 1b5 is therefore met.

696. The dispatcher (CGI Messenger) transmits the request to the pager server (application) in Fig. 2. Limitation 1b6 is therefore met.

697. Popp et al. does not expressly discuss concurrent processing. However, it is found at page 2 of reference 350 and page 3 of reference 348. Limitation 1c is therefore met by the combination.

698. The entire Popp et al. patent is concerned with dynamically generating a web page in response to a request. Limitation 1d1 is therefore met.

699. Retrieving data from an external source and using it in the generated page is in claim 8 of Popp et al. Limitation 1d2 is therefore met.

700. Therefore, '554 claim 1 is obvious in view of Popp et al. reference 194 and the NeXT references 354.

701. Identification of an appropriate external data source is described at 4:34-41; 5:63-6:6. Therefore the limitation of claim 2 is met.

702. The data is dynamically retrieved from an external data source as described at 4:34-41; 5:63-6:6. Therefore the limitation of claim 2 is met.

703. Popp et al. 6,651,108 does not discuss connection caching. However, it is described, but not named as such, in Reference 348 at 9:48-55; 13:28-40; 14:49-15-8. Further, when multiple connections were maintained, connection caching was obvious to those skilled in the art, as explained earlier. Therefore, the limitation of claim 4 is met by the combination.

704. Logging into data sources is not expressly discussed in Popp et al. However, it is described in Reference 248 at 13:29-40. The client's login password is retained so repeated entry of the password is not necessary. Furthermore, as explained above, it would have been obvious to one skilled in the art that access to external data sources was controlled and the presentation of authorization credentials (logging in) would normally be expected. The limitation of claim 5 is therefore met by the combination.

705. Page caching is not expressly discussed in Popp et al. It is described, but not named as such, in reference 348 at 10:19-28. Furthermore, at the time of the Patents and as discussed above, servers commonly cached pages to avoid having to retrieve or generate them

repeatedly, so it would have been obvious to one of skill in the art to cache generated pages. The limitation of claim 6 is therefore met by the combination.

706. Custom HTML extension templates, and the insertion of data into the templates, are described at 3:22-27; 4:34-41; 7:29-58. The limitation of claims 7 and 8 are therefore met.

707. Popp et al. describes a networked system for managing a dynamic page generation request. Therefore limitation 9a is met.

708. Popp et al. describes data sources at 4:34-41; 5:63-6:6. Therefore limitation 9b is met.

709. Popp et al. teach page server (application) 214 running on a computer, which is the requires processing means. Therefore limitation 9c is met.

710. The "first computer system including means for generating said request" is the computer on which the client browser 202 runs. The request is generated by that browser. Therefore limitation 9d is met.

711. The "second computer system" is the one on which the HTTP server (web server) 206 operates. Therefore limitation 9e1 is met.

712. The "means for receiving said request from said first computer" is the HTTP server (web server) 206 running on its computer. Therefore limitation 9e2 is met.

713. The "router, said router routing said request from said second computer system to said page server" is CGI messenger 210. Therefore limitation 9e3 is met.

714. "[S]aid routing further includes intercepting said request at said second computer" occurs when the HTTP server determines that it should not process the request but instead should hand it over to the page server through the CGI messenger 210. Therefore limitation 9e4 is met.

715. "[R]outing said request from said second computer to a dispatcher" occurs when the second computer invokes the CGI messenger to dispatch the request to the page server. Therefore limitation 9e5 is met.

716. "[D]ispatching said request to said page server" occurs when the CGI messenger (dispatcher) 210 transfers the request to the application (pager server) 214. Therefore limitation 9e6 is met.

717. For "said page server receiving said request," see the discussion of limitation 1b2.

718. "[R]eleasing said second computer system to process other requests" is not expressly taught in Popp et al. However, such releasing is described on p. 2 of reference 350. Limitation 9e8 is therefore met by the combination.

719. The entire Popp et al. patent is concerned with dynamically generating a web page in response to a request. Limitation 9e9 is therefore met.

720. Retrieving data from an external source and using it in the generated page is claim in claim 8 of Popp et al. Limitation 9e10 is therefore met.

721. Therefore, claim 9 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

722. '554 claim 10 depends from claim 9. Limitation 10a is met in combination because claim 9 is met by the preceding combination

723. The "interceptor intercepting said request at said second computer system and routing said request" is the code in the HTTP server that determines whether to send the request to CGO messenger 210. Limitation 10b is therefore met by the combination.

724. The "dispatcher receiving said routed request from said interceptor and dispatching said request to said page server" is the CGI messenger 210. Limitation 10c is therefore met by the combination.

725. Therefore, claim 10 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

726. '554 claim 11 is drawn to machine-readable medium. Such a medium is present in Popp et al. since it stores the instructions executed by the server computer system.

727. The functions 11b1-11b6 correspond to process steps 1b1-1b6. Therefore these limitations are met by Popp et al. in combination.

728. Functions 11c and 11d correspond to process steps 1c and 1d1, 1d2 together. Therefore these limitations are met by Popp et al. in combination.

729. Therefore, claims 1-11 of the '554 Patent are obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

730. As to '335 claim 1, steps 1a, 1b1-1b3, 1c1-1c2, 1d, 1e1-2 are analogous to '554 claim 11 steps 11a, 11b1-1b4, 11b6 and 11d, all of which were shown to be present in Popp et al. in combination with the cited references. Therefore, '335 claim 1 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

731. Step 2a of claim 2 is met because routing occurs in Popp et al. as discussed above in connection with limitation 1b1 of '554. Steps 2b-2c of claim 2 correspond to steps 1b5 and 1b6 of '544 claim 1. Therefore, '335 claim 2 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

732. The step of claim 3 corresponds to the step of '554 claim 2. Therefore, '335 claim 3 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

733. The step of claim 4 corresponds to the step of '554 claim 3. Therefore, '335 claim 4 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

734. The step of claim 5 corresponds to the step of '554 claim 4. Therefore, '335 claim 5 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

735. The step of claim 6 corresponds to the step of '554 claim 5. Therefore, '335 claim 6 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

736. The step of claim 7 corresponds to the step of '554 claim 6. Therefore, '335 claim 7 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

737. The step of claim 8 refers to "tag-based text templates." HTML templates are "tag-based text templates," and are present in Popp et al. as shown in the discussion of the limitation in '554 claim 7. Therefore, '335 claim 8 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

738. The step of claim 9 refers to inserting data from data sources into "tag-based text templates." The same insertion into "HTML templates" occurs in Popp et al., as shown in the discussion of the limitation in '554 claim 8. Because "HTML templates" are "tag-based text templates," this limitation is met by Popp et al. Therefore, '335 claim 9 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

739. In Popp et al. the HTML templates determine a format of the data retrieved from data sources. Their purpose is to serve as placeholders and format determiners, as explained at "A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." (2:26-28) See also 3:23-38; 4:28-41; 5:63-6:1; 7:21-58. Therefore, '335 claim 10 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

740. The limitation of claim 11 is met perforce because the limitation of '554 claim 7 is met. Therefore, '335 claim 11 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

741. Claim 12 of '335 requires "dynamically updating data at said one or more data sources." This is not taught expressly by Popp et al. 6,651,108. However, claim 12 of Popp et al. 6,249,291 recites "computer readable program code configured to cause a computer to update said external data source with said object oriented object data." The update capability is also taught at 9:48-55. Therefore, '335 claim 12 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

742. Claim 13 of '335 requires "processing an object handling extension." It is straightforward to locate such a teaching in Popp et al. since that patent deals with "generating object-oriented world wide web pages," as its title proclaims. The object handling extension mechanism is described at 3:53-4:41. Therefore, '335 claim 13 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

743. Claim 14 restricts the object handling extension to be an "OLE extension." Popp et al. 6,651,108 does not mention OLE expressly. However, OLE is described in references 345

and 350. Furthermore, as discussed above, the premier object handling extension at the time of the Patents was Microsoft OLE, and it would have been obvious to one skilled in the art to use OLE when an object handling extension was called for. Therefore, '335 claim 14 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

744. Popp et al. describes a computer implemented method. Therefore, limitation 15a is met.

745. Limitation 15b1, "transferring a request from an HTTP-compliant device to a page server," is met by Popp et al. because the HTTP server 206 is HTTP-compliant and "transferring" and "routing" have the same meaning. Since Limitation 1b1 of '554 is met by Popp et al., so is limitation 15b1 of '335.

746. Limitation 15b2 is identical to limitation 1b2 of '554, hence is met by Popp et al.

747. Limitation 15b3, "releasing said HTTP-compliant device to process other requests" is a general case of limitation 1b3 of '554, hence is met in combination by Popp et al. in combination with the references cited.

748. Limitation 15c, "intercepting said request at said HTTP-compliant device and transferring said request to said page server" is a general case of limitation '335 1c1 and 1c2, hence is met by Popp et al.

749. Limitations 15d1 and 15d2 together are a general case of '335 limitation 1d, hence are met by Popp et al. in combination with the references cited.

750. Limitation 15e is essentially identical to limitations 1d1 and 1d2 of '554, hence is met by Popp et al.

751. Therefore, '335 claim 15 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

752. Limitation 16a is perforce met by Popp et al. in combination with the references cited.

753. Limitation 16b is a general case of limitation 1b5 of '554, hence is met by Popp et al.

754. Limitation 16c is identical to limitation 1b6 of ’554, hence is met by Popp et al.

755. Therefore, ’335 claim 16 is obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

756. The limitation of claim 17 is essentially identical to the limitation of ’554 claim 2, hence is met by Popp et al.

757. The limitation of claim 18 is essentially identical to the limitation of ’554 claim 3, hence is met by Popp et al.

758. The limitation of claim 19 is essentially identical to the limitation of ’554 claim 4, hence is met by Popp et al. in combination with the cited references.

759. The limitation of claim 20 is essentially identical to the limitation of ’554 claim 5, hence is met by Popp et al. in combination with the cited references.

760. The limitation of claim 21 is essentially identical to the limitation of ’554 claim 6, hence is met by Popp et al. in combination with the cited references.

761. The limitation of claim 22 is essentially identical to the limitation of ’335 claim 8, hence is met by Popp et al.

762. The limitation of claim 23 is essentially identical to the limitation of ’335 claim 9, hence is met by Popp et al.

763. The limitation of claim 24 is essentially identical to the limitation of ’335 claim 10, hence is met by Popp et al.

764. The limitation of claim 25 is essentially identical to the limitation of ’335 claim 11, hence is met by Popp et al.

765. The limitation of claim 26 is essentially identical to the limitation of ’335 claim 12, hence is met by Popp et al. in combination with the cited references.

766. The limitation of claim 27 is essentially identical to the limitation of ’335 claim 13, hence is met by Popp et al.

767. The limitation of claim 28 is essentially identical to the limitation of ’335 claim 14, hence is met by Popp et al. in combination with the cited references.

768. Limitation 29a is met by Popp et al. because it is a computer-implemented method.

769. Limitation 29b is essentially identical to limitation 16b of '335, hence is met by Popp et al.

770. Limitation 29c is described in Popp et al. 6,249,291 at 8:54-59, which teaches the use of control objects to "facilitate the flow of information to and from external data sources." It is taught at 9:48-55 to build a database query statement in realtime to retrieve information from an external data source. Part of the query is the location of the data. Hence limitation 29c is met by Popp et al. in combination with Popp et al. 6,249,291.

771. Limitation 29d1 requires "dispatching said request to an appropriate page server based on said request and based on said dynamic information." Popp et al. 6,249,291 teaches using shared objects "across multiple applications executing on application server 316 (or any other server)." (4:26-34) This implies that multiple page servers (applications) can be running on multiple servers. The decision as to which page server (application) should be invoked is based on the "app_name," as described at 7:7-12. Hence limitation 29d1 is met by Popp et al. in combination with Popp et al. 6,249,291.

772. Limitation 29d2 is a general case (HTTP-compliant device) of limitations 9e7 and 9e8 of '554, hence is met by Popp et al. in combination with the cited references.

773. Limitations 29e1 and 29e2 together are a general case (HTTP-compliant device) of limitations 1b6 and 1c of '554, hence are met by Popp et al. in combination with the cited references.

774. Limitation 29f1 is identical to limitation 1d1 of '554, hence is met by Popp et al.

775. Limitation 29f2 is essentially identical to limitation 1d2 of '554, hence is met by Popp et al.

776. Therefore, claims 1-29 of the '335 Patent are obvious in light of Popp et al. 6,651,108, 6,249,291 and the NeXT references.

INVALIDITY PURSUANT TO 35 U.S.C. § 112

Failure to Enable the Full Scope of the Asserted Claims/Lack of Adequate Written Description

777. I understand that a patent claim is invalid pursuant to 35 U.S.C. § 112 ¶ 1 if the patent's Specification fails to "contain a written description of the invention, and of the manner and process of making and using it, in such full, clear concise and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use" it without undue experimentation. I further understand that the so-called "enablement" and "written description" requirements require the specification to contain enabling disclosure for the full scope of the claim in question as asserted by the patentee to be infringed.

778. In my opinion, the specifications of the Patents fail to meet these requirements in several respects.

779. The first such respect relates to epicRealm's infringement contention that multi-threaded software running on a single physical hardware device can infringe the Asserted Claims. As an initial matter, if this contention is correct, none of the Asserted Claims would be valid because they would be anticipated by numerous prior art references, including systems that suffer from the very efficiency problems the Applicants claim to have solved. In addition, far from containing enabling disclosure for a claim that would support such a contention, the express words of the Specification directly contradict epicRealm's attempt to apply the claims this broadly.

780. The Specification states, "Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this

need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers." ('554 4:39-53) There could not be a clearer statement that the invention solves the problem by not using a single machine, but a "partitioned architecture" that must comprise more than one machine. Thus, rather than supporting the multi-threaded architecture asserted by epicRealm, the Specification teaches against the use of a single machine.

781. Indeed, there is no embodiment disclosed in which all elements reside on a single machine, nor could there be. Such an embodiment would be anticipated because it is shown as prior art in Figures 2 and 3 of the Specification. Such an embodiment would also fail to achieve the advantages set forth by Applicants.

782. Further teaching against a single machine can be found at '554 6:14:19: "In this scenario, Dispatcher 402 can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This 'load balancing' capability can significantly increase performance at a busy Web site." Such a teaching makes no sense and is incorrect if the page server and web server reside on the same machine. A machine cannot "load balance" itself, since there is no other place for requests to be sent, and no such balancing could occur. Such a single-machine configuration, could become "highly inefficient," defeating the purported purpose of the claimed invention.

783. Thus, no embodiment of the invention running on a single machine is enabled by the Specification. Given the exposition by Applicants at '554 4:39-53, it would be impossible to realize the benefit of the invention on a single machine, and the Specification gives no instruction on how to make and use the invention with a single machine. It is in fact logically impossible to do so. A single machine can execute only a given number of machine instructions per second, and if time is spent on generating dynamic pages it cannot be spent on servicing other requests.

784. Therefore, if epicRealm contends that any of the Asserted Claims can be infringed by a single machine, that Asserted Claim is invalid both for anticipation and non-enablement.

785. A second failure to provide enabling disclosure for certain of epicRealm's Asserted Claims relates to the certain claims of the'335 Patent, which generically claim a method not enabled by the Specification. All of the embodiments disclosed in the Specification teach an interceptor that intercepts a request and routes it to a dispatcher, which then receives the request and dispatches it to a chosen page server. However, independent claims 1, 15 and 29 of the '335 Patent are drawn to a generic method that does not incorporate any intercepting or dispatching. The same is true for dependent claims 2, 4-14 and 17-28, which depend from the independent claims referenced above.

786. Even if those claims were determined to cover systems employing intercepting and dispatching steps or devices, those steps or devices are not required by the language of the claims, and those claims have not been construed to include such a requirement. Thus, in order for those claims to be valid, the Specification must provide enabling disclosure for the full scope of the claims, which would include a system or method in which the claimed invention is implemented without intercepting or dispatching.

787. The Specification contains no such disclosure. The only diagram showing the configuration of the claimed invention is Figure 4, which clearly shows Interceptor 400 and Dispatcher 402 as being part of the invention. The only flow chart disclosing the patented invention is Figure 5. Eight of the eleven steps depicted in that flow chart are performed by either the Interceptor, the Dispatcher, or both. The textual description of the patented invention similarly requires the Interceptor and Dispatcher. (4:43-47; 5:64-65) There is absolutely no disclosure of any method or system that practices the claimed invention without the use of an Interceptor and Dispatcher, and in my opinion, one of ordinary skill in the art could not practice such an implementation without undue experimentation since even as of today it is not known in the field how to do so. The Specification therefore does not teach one of skill in the art how to practice the full scope of the invention claimed in the cited claims and maintaining those claims would grant to Applicants that which they did not invent.

Lack of Adequate Written Description of "Releasing" and "Intercepting" Terms

788. In addition to the deficiencies described above concerning enablement of the full scope of the claims, the Specifications of the '554 and '335 Patents also fail to provide adequate written description for the terms "releasing" and "intercepting."

789. The Texas Court has construed the limitation "said page server receiving said request and releasing said Web server to process other requests" to mean "said page server receiving said request and said page server performing an act (separate from merely receiving the request) to free the Web server to process other requests." In this action, epicRealm has proposed that "releasing" should be construed to mean "freeing," but this is no different because it still requires the page server to perform some overt act to "free" the Web server to process other requests. Neither the '554 Patent nor the '335 Patent contains any disclosure whatsoever of any such "act" performed by any page server that frees a Web server to process other requests.

790. In the '554 Patent, "releasing" appears only in the title, the abstract, the claims and the following summary statement: "In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests." (2:21-26). The latter is simply a recitation of what is claimed, without any description of the "releasing" portion of the claimed invention or of the manner and process of making and using it.

791. In the '335 Patent, "releasing" appears in the abstract (not the title), the claims and the same summary sentence that appeared in the '554 Patent. There is no other mention of releasing. Thus, like the '554 Patent, the '335 Patent contains no written description at all of any "act" by a page server that releases the Web server to process other requests. All asserted claims are therefore invalid pursuant to 35 U.S.C. §112 for this reason as well.

792. Adequate written description of the term "intercepting" is also absent from the Specifications of the Patents. All the Asserted Claims except claim 29 of the '335 Patent recite

"intercepting." However, there is no written description of how intercepting is accomplished. Even if one assumed that only dynamic Web page generation requests were to be intercepted, there is no teaching of how such requests are to be distinguished from all other requests.

793. There is not even an unambiguous disclosure of what the interceptor is or where it may be found. Fig. 4 shows the interceptor 400 as lying between the Web server executable 201 and the dispatcher 402. But if that is the arrangement, then the interceptor cannot possibly intercept anything before it gets to the Web server executable, as claimed. In the arrangement of Fig. 4, if the web server must determine that the request is to be passed to the interceptor (i.e. that the request is for a dynamic page), then there is no further work for the interceptor to do since the intercepting function would already have been performed by the web server itself. One of skill in the art would have no idea how to implement the system drawn in Fig. 4 in a manner consistent with the teachings of the Patents. Likewise, Fig. 5 states that intercepting 504 occurs after the receiving 502 of the request by the Web server. This flies in the face of the usual meaning of intercepting, which is deflecting something before it reaches its destination. If Applicants were trying to use "intercepting" in other than its ordinary sense, they failed to define it.

794. The Specification later states that "in an alternate embodiment, Interceptor 400 is an independent module, connected via an "intermediate program" to Web server 201. This intermediate program can be a simple CGI application program that connects Interceptor 400 to Web server 201." (5:1-5). There is no disclosure or description whatsoever of what this "intermediate program" might be or what it should do. It is not shown in any drawing and is never described. One of skill in the art would have no idea, following the plan of Fig. 4, how to implement such an intermediate program, since it would have no function to perform. By the time the decision is made to send a request from the Web server to the interceptor (i.e. for a dynamic page generation request), intercepting has already occurred. If it hasn't yet occurred, and the request is not for a dynamic page, the interceptor would have to send the request back to the Web server for processing, and there is no teaching or suggestion of such a flow in the

Patents. Therefore, there is no written description of this alternate embodiment. Because the Patents' Specifications contain no written description of the "intercepting" term, the claims in what that term appears are invalid for that reason as well.

795. As shown in '554 Fig. 4, the page server does not even communicate directly with the web server 201, but goes through an interceptor and dispatcher 402. It is not clear how the page server could explicitly release the web server in such a configuration. The releasing act would have to be propagated back through the dispatcher 402 and interceptor 400 to have an effect on web server 201. No such back-propagation of releasing is disclosed in the Patents. Fig. 5 is a flowchart of the processing that occurs in generating a web page. There is no block corresponding to "releasing." If releasing were taught in the disclosure, one would expect an "releasing" box between boxes 516 and 518 in Fig. 5, but none is there.

796. Every claim of both Patents requires the page server to perform the act of releasing the web server (or HTTP-compliant device). Fig. 4 of the Patents shows that the page server does not communicate directly with the web server, but goes through at least a dispatcher. (It is ambiguous in Fig. 4 whether the interceptor is part of the web server.) If it is not part of the web server then the effect of the releasing act must also pass through the interceptor. There is no disclosure or written description of the interceptor or the dispatcher releasing the web server, or of either or both of these elements propagating any releasing act that might be performed by the page server.

797. Indeed, none of epicRealm's theories of releasing makes sense in light of Fig. 4. If either TCP releasing or SBL releasing occurs, then the web server releases the dispatcher, not the web server, and there is no suggestion anywhere in the patents that the dispatcher ever releases the web server.

798. Claims 13 and 27 of the '335 Patent are drawn to the generic "object handling extension," while the only embodiments disclosed teach Online Linking and Embedding (OLE) 2.0 as the extension mechanism. The generic claims 13 and 27 are invalid because they are drawn to an embodiment that is not disclosed. Furthermore, the term "object handling

extension" has no ascertainable unambiguous meaning to one of skill in the art, so the claims in which it appears are invalid. One of skill in the art is not taught how to utilize any object handling extension other than OLE 2.0, and there is no suggestion that any other object handling extension might be used or how that might be done.

Secondary Considerations

799. In Defendant's First Supplemental Response to Plaintiffs' Interrogatory Nos. 4 and 12, epicRealm asserts that the inventions of the Patents are not obvious based on certain secondary considerations, specifically commercial success "as demonstrated by the quantity of infringing sales, including but not limited to, the billions of dollars of sales of Oracle's accused products" and a "long felt need to more efficiently manage dynamic page generation requests to a webserver," (First Supplemental Response, p. 4). epicRealm also alleges "failed attempts by others to find a solution to the problems that are solved by the patent-in-suit" and copying by others in the industry. (First Supplemental Response, p. 5). epicRealm further alleges that the claimed inventions were licensed to others and that "it was entirely unexpected that the problems discussed above are solved by the patents-in-suit could be solved by a method, a networked system, or a machine-readable medium that examined a request to make an informed decision as to which page server should perform the request." (First Supplemental Response, p. 6).

800. Other than licensing by others, which is a factual matter, epicRealm's allegations are simply a conclusory catalog of known types of secondary considerations and these allegations are completely controverted by the published prior art.

801. Any claim that the commercial success of Oracle's products might result from the inventions of the Patents I can only characterize as ludicrous. Oracle was the first major software company to announce a comprehensive Internet strategy, in 1995[11]. Oracle 1.0, which anticipates all of the Asserted Claims and was part of that strategy, predates the earliest date of conception claimed by epicRealm. It was available to the public at least as early as September

[11] From a historical timeline published in "Profit: The Executive's Guide to Oracle Applications," May 2007. ORCL01818200-8207.

1995. Second, Oracle's software revenue for 1995 was already $2.6 billion, which placed it among the top three software companies in the world[12].

802. Any "long-felt need" for efficient management of dynamic web page generation requests, if it existed, was amply met by the numerous prior art systems listed in Exhibit 3. That the need was met is apparent from epicRealm's own lack of commercial success in marketing products based on the Patents, which tends to show that any demand in the trade was already being satisfied.

803. While epicRealm alleges that others failed to solve the problems addressed by the Patents, the truth is that those problems had already been solved at the time the alleged inventions were made. Oracle 1.0, for example solved them.

804. The contention that the method of solution disclosed in the Patents was in some way unexpected is contradicted by the fact that multiple prior art systems employed precisely the same process for solving precisely the same problem. This is evident from the similarity of Figure 4 of the Patents to numerous illustrations in this report from prior art systems and the claim element mappings of Exhibit 3.

Failure to Disclose the Best Mode

805. I understand that in order to be valid, a patent must disclose the best mode contemplated by the inventor of carrying out his invention. I further understand that Oracle may assert, based on deposition testimony from named inventor Andrew Levine (the "Levine Depo."), that the inventors of the '554 and '335 patents employed a particular method for practicing certain limitations of the Patents' claims without disclosing that method in the Patents.

806. Mr. Levine testified to the use of a particular algorithm by the dispatcher in the embodiment that epicRealm contends practiced the claimed invention. Mr. Levine referred to that algorithm as "score request," in which a numerical weight was calculated for each page server and the one whose score was best would receive the request. Levine Depo. 69:25-70:13. I have carefully reviewed the entire text of the Patents and have found no disclosure whatsoever

[12] Software Magazine, July 1996. ORCL01818279-8331.

of any algorithm for performing the dispatching function, let alone the "score request" algorithm described by Mr. Levine. Instead, the patents merely state that the "[d]ispatcher 402 receives the intercepted request and then dispatches the request to one of a number of page servers 404(1)-(n)." In my opinion, to the extent that the algorithm about which Mr. Levine testified is determined to constitute the "best mode" for performing the claimed "dispatching" step, known to the inventors at the time of filing, it is not disclosed in the Patents as required.

CONCLUSIONS

807. All of the Asserted claims are anticipated by multiple prior art references.

808. The Oracle 1.0 References anticipate '554 claims 1-3, 5-11 and '335 claims 1-4, 6-12, 15-18, 20-26 and 29.

809. The Oracle 2.0 References anticipate all claims of the Patents.

810. All of the Asserted Claims are obvious in light of vast numbers of combinations of prior art references.

811. None of the Asserted Claims satisfies 35 U.S.C. §112.

812. None of the Asserted Claims is valid.

813. Applicants did not invent dynamic page generation.

814. Applicants did not invent page servers.

815. Applicants did not invent routing.

816. Applicants did not invent intercepting.

817. Applicants did not invent dispatching.

818. Applicants did not invent connection caching.

819. Applicants did not invent logging into databases.

820. Applicants did not invent page caching. Further, applicants did not invent caching of dynamic pages.

821. Applicants did not invent HTML templates or custom HTML extension templates.

822. Applicants did not invent tag-based text templates.

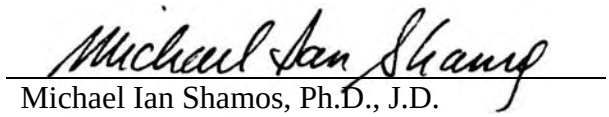
823. Applicants did not invent object handling extensions.

824. Applicants did not invent Object Linking and Embedding (OLE).

825. Applicants did not invent HTTP-compliant devices.

826. Applicants did not invent any of the claimed combinations of the foregoing to respond to dynamic page generation requests.

Executed on May 2, 2008, in Pittsburgh, PA.



Michael Ian Shamos, Ph.D., J.D.

Exhibit 4

	A	B	C	D	E	F	G	H	I	J
1		INVALIDITY ANALYSIS OF LOWERY ET AL. '554 AND '335								
2		NOTE: A large number of cells in this table are automatically replicated. These are denoted by tan shading. Do not attempt to edit these cells								
3		Claim elements found in an accused instrumentality are indicated by non-blank entries in the right-hand portion of the table								
4		Citations to prior art are exemplary only and not exhaustive								
5		A green box means that the corresponding element or step is NOT found in the reference								
6										
7										
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
14					'554					
15					'554	A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of	routing said request from said Web server to a page server	said page server receiving said request and	releasing said Web server to process other requests	wherein said routing step further includes the steps of intercepting said request at said Web server
16	Texas #	**PRIOR ART REFERENCES**	EFF. DATE		'554					
17										
18	018	Poole et al. U.S. Patent 5,455,948, "Method and System For Providing Data To An Application Program"	Dec. 17, 1991		'554	The invention is a pre-Web system for creating and serving dynamic hypertext pages. "The method and system of the present invention provide a way for the user interface component of an application program to operate independent of data and the source of the data. The present invention achieves this result in the context of a script processor. The data retrieval/update method of the present invention is separate from both the using interface component and its associated logic. Therefore, the present invention permits moving the source of data without changing the user interface component." 3:24-32. The server is not a Web server but is the equivalent. The pages are not in HTML, which had not been released yet, but are in hypertext	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	The data collector must receive the request to process it. See also '554 1b1	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	Requests for data are intercepted at server application 148 when a data collector script is processed
19	196	Ng, "GSQL - A Mosaic-SQL Gateway"	Dec. 1993		'554	GSQL is a system for generating HTML pages based on access to an SQL database. p. 2. GSQL is invoked as a script through an HTTP (web) server. p. 1.	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	There is express reference to receiving on p. 2 of the section "GSQL-ORACLE Backend."	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by "name." (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL

	K	L	M	N	O	P	Q
1							
2							
3							
4							
5							
6							
7							
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
14							
15	routing said request from said Web server to a dispatcher	dispatching said request to said page server	processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests	dynamically generating a Web page in response to said request	said Web page including data dynamically retrieved from one or more data sources	wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data	wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources
16							
17							
18	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	Concurrency is inherent in the architecture of Fig. 3. The server application 148 continues to run while the resources 150 are retrieving data	The page is dynamically generated with data from the resources 150 obtained by data collector 102	See '554 1d1	The script identifies the data source(s): "The building block of data collection script files are view definitions which describe data locations and relationships, and command sequences required to interact with the data." 5:35-39	Data is dynamically retrieved from data sources when a data collection script is executed
19	GSQL is the dispatcher. It passes the request to a database backend (page server)	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. “Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command.” (Section “GSQL in Detail,” p. 2).	The HTTP server, which is not processing the web page generation request, is free to process other requests and does so	The Database Backend generates the web page dynamically on request	GSQL accesses a database to obtain data and “gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved	The identification is made from information contained in a PROC file	See '554 1d2

	R	S	T	U	V	W	X	Y
1								
2								
3								
4								
5								
6								
7								
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
14								
15	wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources	wherein said step of processing said request includes the step of logging into said one or more data sources	wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page	wherein said page server includes custom HTML extension templates for configuring said Web page	wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates	A networked system for managing a dynamic Web page generation request, said system comprising	one or more data sources	a page server having a processing means
16								
17								
18						The disclosed invention is networked: "The preferred embodiment uses a server and client workstation connected through a local area network (LAN) with the server running under OS/2 or UNIX operating system and the client workstations running under Windows on a DOS machine or Presentation Manager on an OS/2, UNIX, or Novell machine." 11:61-12:5	See '554 1d1	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18
19		Logging into a Sybase database is taught expressly. See GSQL PROC File Commands, p. 1.		GSQL PROC files are HTML extension templates. "In the meantime, just experiment with the syntax and commands of the PROC file. And it's like a higher-level HTML with its pros and cons." (GSQL PROC file commands, p. 1)	"The $ (dollar) sign is a placeholder where the user-inputs will be substituted directly into." (GSQL PROC file commands, p. 2)	GSQL is networked. "GSQL is a simple Mosaic gateway to SQL databases." (GSQL in detail, p. 1) Mosaic is a Web browser	GSQL accesses a database to obtain data and "gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend

	Z	AA	AB	AC	AD	AE	AF	AG
1								
2								
3								
4								
5								
6								
7								
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
14								
15	a first computer system including means for generating said request	a second computer system including	means for receiving said request from said first computer	said second computer system also including a router, said router routing said request from said second computer system to said page server	wherein said routing further includes intercepting said request at said second computer	routing said request from said second computer to a dispatcher, and	dispatching said request to said page server	said page server receiving said request and
16								
17								
18	The first computer system includes the client 146 of Fig. 3	The second computer system includes the server 148 of Fig. 3	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	Requests for data are intercepted at server application 148 when a data collector script is processed	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	The data collector must receive the request to process it. See also '554 1b1
19	The first computer system is the client running a Mosaic browser	The second computer system includes GSQL and the Database Backends	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL	GSQL is the dispatcher. It passes the request to a database backend (page server)	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. “Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command.” (Section “GSQL in Detail,” p. 2).	There is express reference to receiving on p. 2 of the section “GSQL-ORACLE Backend.”

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
1									
2									
3									
4									
5									
6									
7									
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
14									
15	releasing said second computer system to process other requests	said page server processing means processing said request and dynamically generating a Web page in response to said request	said Web page including data dynamically retrieved from said one or more data sources	The networked system in claim 9 wherein said router in said second computer system includes:	an interceptor intercepting said request at said second computer system and routing said request	a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server	A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of	routing a dynamic Web page generation request from a Web server to a page server	said page server receiving said request and
16									
17									
18	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	The page is dynamically generated with data from the resources 150 obtained by data collector 102	See '554 1d1	The disclosed invention is networked: "The preferred embodiment uses a server and client workstation connected through a local area network (LAN) with the server running under OS/2 or UNIX operating system and the client workstations running under Windows on a DOS machine or Presentation Manager on an OS/2, UNIX, or Novell machine." 11:61-12:5	Requests for data are intercepted at server application 148 when a data collector script is processed	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	Because the system is computerized, its instructions must be on a machine-readable medium	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	The data collector must receive the request to process it. See also '554 1b1
19	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by “name.” (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The Database Backend generates the web page dynamically on request	GSQL accesses a database to obtain data and “gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved	GSQL is networked. "GSQL is a simple Mosaic gateway to SQL databases." (GSQL in detail, p. 1) Mosaic is a Web browser	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. “Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command.” (Section “GSQL in Detail,” p. 2).	Because the system is computerized, its instructions must be on a machine-readable medium. The system is obtained by downloading the file gsql.tar, which is machine-readable	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	There is express reference to receiving on p. 2 of the section “GSQL-ORACLE Backend.”

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
1										
2										
3										
4										
5										
6										
7										
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
14							'335			
15	releasing said Web server to process other requests	intercepting said request at said Web server	routing said request from said Web server to a dispatcher	dispatching said request to said page server	processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests	dynamically generating a Web page in response to said request	'335	A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of	routing a request from a Web server to a page server	said page server receiving said request and
16							'335			
17										
18	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	Requests for data are intercepted at server application 148 when a data collector script is processed	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	Concurrency is inherent in the architecture of Fig. 3. The server application 148 continues to run while the resources 150 are retrieving data	The page is dynamically generated with data from the resources 150 obtained by data collector 102	'335	The invention is a pre-Web system for creating and serving dynamic hypertext pages. "The method and system of the present invention provide a way for the user interface component of an application program to operate independent of data and the source of the data. The present invention achieves this result in the context of a script processor. The data retrieval/update method of the present invention is separate from both the using interface component and its associated logic. Therefore, the present invention permits moving the source of data without changing the user interface component." 3:24-32. The server is not a Web server but is the equivalent. The pages are not in HTML, which had not been released yet, but are in hypertext	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	The data collector must receive the request to process it. See also '554 1b1
19	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by “name.” (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL	GSQL is the dispatcher. It passes the request to a database backend (page server)	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. “Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command.” (Section “GSQL in Detail,” p. 2).	The HTTP server, which is not processing the web page generation request, is free to process other requests and does so	The Database Backend generates the web page dynamically on request	'335	GSQL is a system for generating HTML pages based on access to an SQL database. p. 2. GSQL is invoked as a script through an HTTP (web) server. p. 1.	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	There is express reference to receiving on p. 2 of the section “GSQL-ORACLE Backend.”

	BA	BB	BC	BD	BE	BF	BG	BH
1								
2								
3								
4								
5								
6								
7								
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
14								
15	releasing said Web server to process other requests	wherein said routing step further includes the steps of intercepting said request at said Web server	routing said request to said page server	processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests	dynamically generating a Web page in response to said request	said Web page including data dynamically retrieved from one or more data sources	The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of:	routing said request from said Web server to a dispatcher
16								
17								
18	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	Requests for data are intercepted at server application 148 when a data collector script is processed	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	Concurrency is inherent in the architecture of Fig. 3. The server application 148 continues to run while the resources 150 are retrieving data	The page is dynamically generated with data from the resources 150 obtained by data collector 102	See '554 1d1	The invention is a pre-Web system for creating and serving dynamic hypertext pages. "The method and system of the present invention provide a way for the user interface component of an application program to operate independent of data and the source of the data. The present invention achieves this result in the context of a script processor. The data retrieval/update method of the present invention is separate from both the using interface component and its associated logic. Therefore, the present invention permits moving the source of data without changing the user interface component." 3:24-32. The server is not a Web server but is the equivalent. The pages are not in HTML, which had not been released yet, but are in hypertext	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source
19	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by “name.” (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	The HTTP server, which is not processing the web page generation request, is free to process other requests and does so	The Database Backend generates the web page dynamically on request	GSQL accesses a database to obtain data and “gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved	GSQL is a system for generating HTML pages based on access to an SQL database. p. 2. GSQL is invoked as a script through an HTTP (web) server. p. 1.	GSQL is the dispatcher. It passes the request to a database backend (page server)

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
1									
2									
3									
4									
5									
6									
7									
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
14									
15	dispatching said request to said page server	wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data	wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources	wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources	wherein said step of processing said request includes the step of logging into said one or more data sources	wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page	wherein said page server includes tag-based text templates for configuring said Web page	wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates	wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request
16									
17									
18	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	The script identifies the data source(s): "The building block of data collection script files are view definitions which describe data locations and relationships, and command sequences required to interact with the data." 5:35-39	Data is dynamically retrieved from data sources when a data collection script is executed				The hypertext is a tag-based text template. "In the example of FIG. 7, client report, hypertext, and chart 176 may require data from a variety of sources. For the needed data, report, hypertext, and chart 176 communicates with interface-to-data-collector dynamic link libraries (DLL's) 220 and to data collectors 102 to reach the various data collection scripts 222 through 232. Each of the data collection scripts 222 through 232 may contain data that the client report, hypertext, or chart 176 needs. In order to access the data collection scripts 222 through 232, the preferred embodiment provides data collectors 102 which may include the SV3270 data collector 234, the FLATFILE data collector 236, SQLDB data collector 238, TTY data collector 240, 914 data collector 242, or the DAA data collector 244." 9:6:19	See '335 8	The purpose of hypertext is to drive a format of the data
19	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. "Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command." (Section "GSQL in Detail," p. 2).	The identification is made from information contained in a PROC file	See '554 1d2		Logging into a Sybase database is taught expressly. See GSQL PROC File Commands, p. 1.		See '554 7	See '554 8.	The templates are used for formatting: "The output will be automatically piped to the Mosaic client. Presumably you will want to do your own HTML formatting, complete with images, links, etc, etc." (GSQL in detail, p. 3)

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
1										
2										
3										
4										
5										
6										
7										
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
14										
15	wherein said tag-based text templates include HTML templates	wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources	wherein said step of processing said request further includes the step of processing an object handling extension	wherein said object handling extension is an OLE extension	A computer-implemented method comprising the steps of:	transferring a request from an HTTP-compliant device to a page server	said page server receiving said request and	releasing said HTTP-compliant device to process other requests	wherein said transferring step further includes the steps of intercepting said request at said HTTP-compliant device and	transferring said request to said page server
16										
17										
18		The invention teaches use of SQL, of which one of the primitives is UPDATE. "The SQLDB data collector retrieves and stores data in an SQL database. The preferred embodiment initially supports Microsoft's SQL server or FairCom SQL server." 6:5-7			The invention is computer-implemented	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18	The data collector must receive the request to process it. See also '554 1b1	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	Requests for data are intercepted at server application 148 when a data collector script is processed	The equivalent of the Web server is that component of server application 148 of Fig. 3 which communicates with the client and listens for requests. The page server is data collector 102 of Fig. 3. The routing is taught as follows: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102. Data collector 102 permits server application 148 to communicate with facilities 150." 7:13-18
19	See '554 7.	"You get to write the routine execute-sql( ). The SQL string is passed into this routine, and what you do with it is totally up to you." (GSQL in detail, p. 3) SQL provides for dynamic update of databases			GSQL is computer-implemented	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend	There is express reference to receiving on p. 2 of the section "GSQL-ORACLE Backend."	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by "name." (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The HTTP server does not process the web page generation request but it is intercepted and passed on to GSQL	The Web server is the HTTP server of the reference. The page server is the "database backend." The request is routed (sent) from the HTTP (web) server to GSQL and ultimately to a database backend

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
1									
2									
3									
4									
5									
6									
7									
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
14									
15	processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests	dynamically generating a Web page in response to said request	said page including data dynamically retrieved from one or more data sources	The computer-implemented method of claim 15 wherein said step of transferring said request includes the steps of:	transferring said request from said HTTP-compliant device to a dispatcher	dispatching said request to said page server	wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data	wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources	wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources
16									
17									
18	Concurrency is inherent in the architecture of Fig. 3. The server application 148 continues to run while the resources 150 are retrieving data	The page is dynamically generated with data from the resources 150 obtained by data collector 102	See '554 1d1	The invention is computer-implemented	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	The script identifies the data source(s): "The building block of data collection script files are view definitions which describe data locations and relationships, and command sequences required to interact with the data." 5:35-39	Data is dynamically retrieved from data sources when a data collection script is executed	
19	The HTTP server, which is not processing the web page generation request, is free to process other requests and does so	The Database Backend generates the web page dynamically on request	GSQL accesses a database to obtain data and "gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved	GSQL is computer-implemented	GSQL is the dispatcher. It passes the request to a database backend (page server)	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. "Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command." (Section "GSQL in Detail," p. 2).	The identification is made from information contained in a PROC file	See '554 1d2	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
1										
2										
3										
4										
5										
6										
7										
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
14										
15	wherein said step of processing said request includes the step of logging into said one or more data sources	wherein said step of dynamically generating said page includes the step of maintaining a page cache containing said page	wherein said page server includes tag-based text templates for configuring said Web page	wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates	wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request	wherein said tag-based text templates include HTML templates	wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources	wherein said step of processing said request further includes the step of processing an object handling extension	wherein said object handling extension is an OLE extension	A computer-implemented method comprising the steps of:
16										
17										
18			The hypertext is a tag-based text template. "In the example of FIG. 7, client report, hypertext, and chart 176 may require data from a variety of sources. For the needed data, report, hypertext, and chart 176 communicates with interface-to-data-collector dynamic link libraries (DLL's) 220 and to data collectors 102 to reach the various data collection scripts 222 through 232. Each of the data collection scripts 222 through 232 may contain data that the client report, hypertext, or chart 176 needs. In order to access the data collection scripts 222 through 232, the preferred embodiment provides data collectors 102 which may include the SV3270 data collector 234, the FLATFILE data collector 236, SQLDB data collector 238, TTY data collector 240, 914 data collector 242, or the DAA data collector 244." 9:6:19	See '335 8	The purpose of hypertext is to drive a format of the data		The invention teaches use of SQL, of which one of the primitives is UPDATE. "The SQLDB data collector retrieves and stores data in an SQL database. The preferred embodiment initially supports Microsoft's SQL server or FairCom SQL server." 6:5-7			The invention is computer-implemented
19	Logging into a Sybase database is taught expressly. See GSQL PROC File Commands, p. 1.		See '554 7	See '554 8.	The templates are used for formatting: "The output will be automatically piped to the Mosaic client. Presumably you will want to do your own HTML formatting, complete with images, links, etc, etc." (GSQL in detail, p. 3)	See '554 7.	"You get to write the routine execute-sql( ). The SQL string is passed into this routine, and what you do with it is totally up to you." (GSQL in detail, p. 3) SQL provides for dynamic update of databases			GSQL is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
1								
2								
3								
4								
5								
6								
7								
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
14								
15	transferring a request from an HTTP-compliant device to a dispatcher	maintaining dynamic information regarding data sources a given page server may access	dispatching said request to an appropriate page server based on said request and based on said dynamic information	said page server receiving said request and	releasing said HTTP-compliant device to process other requests	processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests	dynamically generating a Web page in response to said request	said page including data dynamically retrieved from one or more data sources
16								
17								
18	The dispatcher is script processor 158: "server application 148 operates from OS/2 platform 154 through system monitor 156 and script processor 158 (including job queuing and scheduler facilities 160 and 162, respectively) to data collector 102." In another view, the dispatcher is Server Data Manager 191, which invokes the appropriate data collector 102 to access the data source	The dynamic information is in the data collector scripts which may be dynamically updated	The dispatcher makes an informed selection which data collector to invoke based on the script processed by the script processor 158. Further, the SDM 191 maintains dynamic information about page servers in Queue Daemon 186, which information is used by scheduler 188 to manage data collection	The data collector must receive the request to process it. See also '554 1b1	Communication in the invention is by pipes (for applications on the same processor) or TCP/IP across a network. TCP provides TCP releasing per EpicRealm's contentions. Therefore, the server application 148 is released when the request is handed off	Concurrency is inherent in the architecture of Fig. 3. The server application 148 continues to run while the resources 150 are retrieving data	The page is dynamically generated with data from the resources 150 obtained by data collector 102	See '554 1d1
19	GSQL is the dispatcher. It passes the request to a database backend (page server)	Dynamic information about data sources is kept in PROC files, which can be updated at any time	GSQL determines which page server (Backend) to use based on entries in a PROC file, which may be modified dynamically. Thus it makes an informed selection based on dynamic information about page servers. "Recall that GSQL will build the SQL string, parse the DEFINES from the PROC file, and then call the Backend specified in the PROC file using the SQLPROG command." (Section "GSQL in Detail," p. 2).	There is express reference to receiving on p. 2 of the section "GSQL-ORACLE Backend."	The routine app-getdefine (char * name) must be called and it returns a pointer to the value of the define specified by "name." (GSQL in Detail, p. 3). The return of this pointer indicates that the server may process other requests. The calling thread remains blocked until the pointer is returned. Also, GSQL issues a return code to the calling program (HTTP server) when it parses the request. The receipt of this code indicates to the server that it is free to process other requests.	The HTTP server, which is not processing the web page generation request, is free to process other requests and does so	The Database Backend generates the web page dynamically on request	GSQL accesses a database to obtain data and "gsql will invoke the stand-alone executable ORACLE backend, with the SQL query as the command line argument. The backend will process the query and return the results embedded in HTML. These results are sent back to the user. (GSQL-ORACLE Backend, p. 2). The embedded results are the ones dynamically retrieved

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
20	066	Luotonen, "World-Wide Web Proxies"	Apr. 1994		'554	The reference discloses handing off document requests to remote servers: "When a normal HTTP request is made by a client, the HTTP server ... knows the host machine that it is running on. The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3. CGI scripts are used to generate dynamic documents	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	Fig. 3 shows the page server (HTTP server) receiving the request	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)
21	00H	Dönszelmann et al., “Gateways for World-Wide Web in the ‘Online’ Data Acquisition System of the DELPHI Experiment at CERN"	Oct. 1994		'554	“When a WWW client makes a request for a ‘DIM page’, the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client” (Section “DIM-WWW Unstructured Gateway”).	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	The page server cannot process the request unless it receives it.	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere
22	019	Srinivas, “Applications of Mosaic in Health Care Delivery”	Oct. 1994		'554	The reference discloses a system called ARTEMIS for accessing and updating databases of medical information through a WWW interface. Patient data is stored in “heterogeneous, distributed databases” (p. 3), then retrieved on demand via queries and pages are converted on the fly to web pages using an HTML interface.	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The page server must receive a request to process it	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
20			Concurrency is inherent. See '554 1b3	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client	See '554 1d1	See '554 1d2
21	The dispatcher is the "HIPE-WWW Structured Gateway"	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	Dynamic requests are passed to the HIPE gateway, which leaves the Web server daemon free to accept other requests. This means that the HTTP-compliant device concurrently processes other requests while the page server is processing a page.	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	See '554 1d1. The "data sources" are the services, which provide information for the page	The identification is provided by the "DIM service names," which tell where to retrieve data	See '554 1d2
22	The dispatcher is the Session Manager, which sends the request on to a gateway	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	"The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users." p. 5. The architecture with multiple gateways is inherently concurrent	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	See '554 1d1, "queries to information servers"	The "user defined models" specify this information: "There are a collection of user defined models that specify what types of information that needs to be served by the system." p. 5	See '554 1d2

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
20			The reference teaches page caching expressly: "This makes it possible for the proxy to do efficient caching of documents that are requested by a number of clients." p. 1			The disclosed system is Web-based, hence networked. See also '554 1a	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.
21			A page cache is taught expressly as the DIM-WWW cache: "The cache operates in much the same way as a normal cache, i.e. whenever information from a service is required, the cache is searched to see if it already contains the information. If so, this can be copied directly into the document in place of the <DIM> tag" Sec. "DIM-WWW cache"	The reference describes an HTML extension known as HTML+: "DIM-WWW is an unstructured gateway, in which HTML+ was extended by one extra tag. The gateway replaces this tag with information retrieved from the `Online' system." Abstract	See '554 7	The system of the reference is Web-based, hence networked	See '554 1d1. The "data sources" are the services, which provide information for the page	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: "The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information." (Section "HIPE-WWW Structured Gateway.") The routing is illustrated in the reference
22	The session manager maintains a cache of connections: "The Session Manager instantiates a new session for each user who logs into the system. This process involves instantiating a specific set of gateways (such as Oracle gateway and file archiver), setting up sessions to these as the user who has just logged in and instantiating models (see next section) which interface to these gateways." p. 5	Logging in is handled by the session manager, which also provides access to databases		Artemis agents use HTML+, which is a set of HTML extensions	The insertions are made in "the layout page associated with the document type." p. 4	ARTEMIS is a networked system	See '554 1d1, "queries to information servers"	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
20	The first computer system includes the Client of Fig. 3	The second computer system includes the Proxy server and HTTP server of Fig. 3	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)			Fig. 3 shows the page server (HTTP server) receiving the request
21	The first computer system is shown as the "Browser/Client"	The second computer system includes the "Front-End HTTP Server," HIP, DIM and the "Back End"	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere	The dispatcher is the "HIPE-WWW Structured Gateway"	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	The page server cannot process the request unless it receives it.
22	The first computer system is the "Mosaic Compliant GUI" of the figure on p. 4. The means for generating said request is a Web client running a web browser	The second computer system includes the interface manager, session manager, models and gateways	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway	The dispatcher is the Session Manager, which sends the request on to a gateway	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	The page server must receive a request to process it

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
20	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client	The disclosed system is Web-based, hence networked. See also '554 1a	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)		Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	Fig. 3 shows the page server (HTTP server) receiving the request
21	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	See '554 1d1. The "data sources" are the services, which provide information for the page	The system of the reference is Web-based, hence networked	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	The page server cannot process the request unless it receives it.
22	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	See '554 1d1, "queries to information servers"	ARTEMIS is a networked system	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The page server must receive a request to process it

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
20	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)			Concurrency is inherent. See '554 1b3	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	'335	The reference discloses handing off document requests to remote servers: "When a normal HTTP request is made by a client, the HTTP server ... knows the host machine that it is running on. The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3. CGI scripts are used to generate dynamic documents	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	Fig. 3 shows the page server (HTTP server) receiving the request
21	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere	The dispatcher is the "HIPE-WWW Structured Gateway"	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	Dynamic requests are passed to the HIPE gateway, which leaves the Web server daemon free to accept other requests. This means that the HTTP-compliant device concurrently processes other requests while the page server is processing a page.	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	'335	"When a WWW client makes a request for a 'DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client" (Section "DIM-WWW Unstructured Gateway").	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: "The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information." (Section "HIPE-WWW Structured Gateway.") The routing is illustrated in the reference	The page server cannot process the request unless it receives it.
22	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway	The dispatcher is the Session Manager, which sends the request on to a gateway	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	"The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users." p. 5. The architecture with multiple gateways is inherently concurrent	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	'335	The reference discloses a system called ARTEMIS for accessing and updating databases of medical information through a WWW interface. Patient data is stored in "heterogeneous, distributed databases" (p. 3), then retrieved on demand via queries and pages are converted on the fly to web pages using an HTML interface.	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The page server must receive a request to process it

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
20	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	Concurrency is inherent. See '554 1b3	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client	The reference discloses handing off document requests to remote servers: "When a normal HTTP request is made by a client, the HTTP server ... knows the host machine that it is running on. The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3. CGI scripts are used to generate dynamic documents	
21	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	Dynamic requests are passed to the HIPE gateway, which leaves the Web server daemon free to accept other requests. This means that the HTTP-compliant device concurrently processes other requests while the page server is processing a page.	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	See '554 1d1. The "data sources" are the services, which provide information for the page	“When a WWW client makes a request for a ‘DIM page’, the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client” (Section “DIM-WWW Unstructured Gateway”).	The dispatcher is the "HIPE-WWW Structured Gateway"
22	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	"The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users." p. 5. The architecture with multiple gateways is inherently concurrent	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	See '554 1d1, "queries to information servers"	The reference discloses a system called ARTEMIS for accessing and updating databases of medical information through a WWW interface. Patient data is stored in “heterogeneous, distributed databases” (p. 3), then retrieved on demand via queries and pages are converted on the fly to web pages using an HTML interface.	The dispatcher is the Session Manager, which sends the request on to a gateway

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
20		See '554 1d1	See '554 1d2			The reference teaches page caching expressly: "This makes it possible for the proxy to do efficient caching of documents that are requested by a number of clients." p. 1			
21	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	The identification is provided by the "DIM service names," which tell where to retrieve data	See '554 1d2			A page cache is taught expressly as the DIM-WWW cache: "The cache operates in much the same way as a normal cache, i.e. whenever information from a service is required, the cache is searched to see if it already contains the information. If so, this can be copied directly into the document in place of the <DIM> tag" Sec. "DIM-WWW cache"	See '554 7. HTML+ comprises tag-based text templates	See '554 8	Extended HTML+ control formatting: "Three conversion attributes IN, CONVERT, and OUT are provided. These can be used in any combination to control the formatting and appearance of the information." Sec. "DIM-WWW Implementation"
22	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	The "user defined models" specify this information: "There are a collection of user defined models that specify what types of information that needs to be served by the system." p. 5	See '554 1d2	The session manager maintains a cache of connections: "The Session Manager instantiates a new session for each user who logs into the system. This process involves instantiating a specific set of gateways (such as Oracle gateway and file archiver), setting up sessions to these as the user who has just logged in and instantiating models (see next section) which interface to these gateways." p. 5	Logging in is handled by the session manager, which also provides access to databases		HTML+. See '554 7	HTML+. See '554 7	Layout pages drive formats

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
20					The reference describes computer-implemented methods	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.	Fig. 3 shows the page server (HTTP server) receiving the request	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	The request is intercepted because the Web server (proxy server) is not capable of handling it (not having the required resources)	The Web server is the proxy server of Fig. 3. The page server is the HTTP server of Fig. 3. The routing is shown in Fig. 3.
21	The tags in HTML+ are HTML tags, so HTML+ templates are HTML templates				The reference describes computer-implemented methods	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference	The page server cannot process the request unless it receives it.	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	The request is intercepted at the Front-End HTTP Server and not processed by it, but is routed elsewhere	The Web server (HTTP-compliant device) is the "Front-End HTTP Server." The page servers are HIPE and DIM, as described in the reference: “The HIPE-WWW gateway basically maps URLs onto the HIPE function and menu system. The URL may specify the retrieval of a menu, in which case the gateway will produce a list in HTML. Each element in this list will link to the same gateway specifying the retrieval of a submenu or the execution of a function. The latter will result in a list of channels, each containing specific information.” (Section “HIPE-WWW Structured Gateway.”) The routing is illustrated in the reference
22	HTML+. See '554 7	ARTEMIS provides “wide-area access to patient records and update capabilities to structured and unstructured information.” p. 2. Also, an SQL query is shown on p. 5. SQL provides for dynamic update of databases.	CORBA is object-based. It is accessed through an object handling extension		ARTEMIS is computer-implemented	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure	The page server must receive a request to process it	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	The request is intercepted (not handled) by the Web server but passed on to the session manager, model or gateway	The Web server is the "Interface Manager" of the figure on p .4. The page server includes a "Model" and a "Gateway." The routing is shown in the figure

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
20	Concurrency is inherent. See '554 1b3	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client	The reference describes computer-implemented methods			See '554 1d1	See '554 1d2	
21	Dynamic requests are passed to the HIPE gateway, which leaves the Web server daemon free to accept other requests. This means that the HTTP-compliant device concurrently processes other requests while the page server is processing a page.	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	See '554 1d1. The "data sources" are the services, which provide information for the page	The reference describes computer-implemented methods	The dispatcher is the "HIPE-WWW Structured Gateway"	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	The identification is provided by the "DIM service names," which tell where to retrieve data	See '554 1d2	
22	"The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users." p. 5. The architecture with multiple gateways is inherently concurrent	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	See '554 1d1, "queries to information servers"	ARTEMIS is computer-implemented	The dispatcher is the Session Manager, which sends the request on to a gateway	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	The "user defined models" specify this information: "There are a collection of user defined models that specify what types of information that needs to be served by the system." p. 5	See '554 1d2	The session manager maintains a cache of connections: "The Session Manager instantiates a new session for each user who logs into the system. This process involves instantiating a specific set of gateways (such as Oracle gateway and file archiver), setting up sessions to these as the user who has just logged in and instantiating models (see next section) which interface to these gateways." p. 5

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
20		The reference teaches page caching expressly: "This makes it possible for the proxy to do efficient caching of documents that are requested by a number of clients." p. 1								The reference describes computer-implemented methods
21		A page cache is taught expressly as the DIM-WWW cache: "The cache operates in much the same way as a normal cache, i.e. whenever information from a service is required, the cache is searched to see if it already contains the information. If so, this can be copied directly into the document in place of the <DIM> tag" Sec. "DIM-WWW cache"	See '554 7. HTML+ comprises tag-based text templates	See '554 8	Extended HTML+ control formatting: "Three conversion attributes IN, CONVERT, and OUT are provided. These can be used in any combination to control the formatting and appearance of the information." Sec. "DIM-WWW Implementation"	The tags in HTML+ are HTML tags, so HTML+ templates are HTML templates				The reference describes computer-implemented methods
22	Logging in is handled by the session manager, which also provides access to databases		HTML+. See '554 7	HTML+. See '554 7	Layout pages drive formats	HTML+. See '554 7	ARTEMIS provides “wide-area access to patient records and update capabilities to structured and unstructured information.” p. 2. Also, an SQL query is shown on p. 5. SQL provides for dynamic update of databases.	CORBA is object-based. It is accessed through an object handling extension		ARTEMIS is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
20				Fig. 3 shows the page server (HTTP server) receiving the request	Simultaneous processing is inherent in the architecture, since the Web server and page server are different machines. However, since the request is passed by HTTP, there is TCP releasing according to EpicRealm's contention	Concurrency is inherent. See '554 1b3	A CGI script on the page server generates the dynamic page: "The requested path specifies the document or a CGI script on the local filesystem of the server, or some other resource available from that server." p. 3	"the proxy server has all the information necessary to make the actual request to the remote server specified in the request URL, using the protocol specified in the URL." p. 3. This means that a database query can be handled and data dynamically returned to the Web server to be sent back to the client
21	The dispatcher is the "HIPE-WWW Structured Gateway"	This information is in a configuration file, which can be dynamically updated. See Sec. "HIPE-WWW Implementation"	The "HIPE-WWW Structured Gateway" makes an informed selection where to send the request based on a configuration file. HIPE consists of "a linear set of channels of different types, each of which may contain a different set of information. A configuration file specifies how channels are combined into groups of channels and other groups." Sec. "HIPE-WWW Structured Gateway." See Fig. 3, showing dynamic information about channels, such as status	The page server cannot process the request unless it receives it.	The request is passed to HIPE using HTTP, so TCP releasing occurs, according to EpicRealm's contention	Dynamic requests are passed to the HIPE gateway, which leaves the Web server daemon free to accept other requests. This means that the HTTP-compliant device concurrently processes other requests while the page server is processing a page.	"When a WWW client makes a request for a `DIM page', the DIM-WWW gateway replaces references to DIM service names in the page with the actual information from those services. The resultant page is then sent to the WWW client." Sec. "DIM-WWW Structured Gateway"	See '554 1d1. The "data sources" are the services, which provide information for the page
22	The dispatcher is the Session Manager, which sends the request on to a gateway	This information is maintained in "user-defined models," which may be updated at any time hence are dynamic	"The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to." p. 5. An informed selection must be made to send the request to a page server that is capable of handling it. As shown in the figure on p. 4, the Session Manager must maintain dynamic information about which Gateways 1 through n are connected so it knows which Information Repositories 1 through n are available for responding to the request	The page server must receive a request to process it	The Web server hands off the request using a network (TCP) interface. Therefore there is TCP releasing according to EpicRealm's contention. CORBA interfaces similarly provide acknowledgement of receipt	"The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users." p. 5. The architecture with multiple gateways is inherently concurrent	ARTEMIS generates pages based on user requests: "The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, POPRAS form, referral form), the layout page associated with the document type, and queries to information servers."	See '554 1d1, "queries to information servers"

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
23	00K	Sjölin, "A WWW Front End to an OODBMS"	Oct. 1994		'554	The references teaches a WWW front end to "LINCKS," an Object-Oriented Database Management System (OODBMS). "The prototype generates HTML documents on the fly from information pieces in the database." p. 1. "LINCKS fits very well in as a document or image server on a single site, and in particular the support of multiple views generated from the same information." Sec. 5	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks	The web server (NCSA httpd) is shown as using the "CGI protocol" to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. "To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks
24	00J	Shklar et al., The 'InfoHarness' Information Integration Platform	Oct. 1994		'554	The reference describes a system for generating dynamic web pages by passing requests from an HTTP Gateway (web server) to an InfoHarness Server. The InfoHarness Server generates "a combination of HTML forms and hyperlinks" and passes it back to the browser. pp. 4-5	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
23	The dispatcher is that portion of cgi2lincks which decides where to route the request	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the **key file** file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing.	"The prototype generates HTML documents on the fly from information pieces in the database." p. 1	The dynamic web page includes data dynamically retrieved from the "LINCKS Database Server." p. 6	Cgi2lincks uses a key file to identify the appropriate w3lincks server, which accesses data sources	See '554 1d2
24	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	"The InfoHarness server processes requests…" and the system provides "support for multiple clients" where the InfoHarness server processes the request while the HTTP Gateway is free to process other requests at the same time. MAC004527-28	The HTTP Gateway "parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." MAC004527	The InfoHarness Server obtains data from relational databases and "IHOs," or InfoHarness objects. "IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.	"IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The index is used to make the identification.	After the InfoHarness server retrieves the data, the HTTP Gateway "parses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." p.7

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
23		“When starting w3lincks, it connects to a LINCKS database and login as a normal user.” p. 6		The General Presentation Descriptor (GPD) contains custom HTML extension templates. “The HTML is generated from actions specified in the FORMATX part of the GPD for each item, terminal or non-terminal (as defined in the STRUCTURE part). The actions are written in a small application language using the parser included in the CIM layer.” p. 5	Sec. 4.3 describes the insertion in detail	LINCKS is a networked system	The dynamic web page includes data dynamically retrieved from the “LINCKS Database Server.” p. 6	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server
24						InfoHarness is networked. It "supports the largely automatic generation of InfoHarness repositories, and provides access to the original information from Mosaic and other World- Wide Web (WWW) browsers through an HTTP gateway." p. 2.	The InfoHarness Server obtains data from relational databases and “IHOs,” or InfoHarness objects. “IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness.” p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
23	The first computer system is a client workstation. p. 2	The second computer system includes the httpd, cgi2lincks and w3lincks servers as shown in Fig. 6	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks	The dispatcher is that portion of cgi2lincks which decides where to route the request	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks
24	The first computer system is shown in Fig. 1 as an "IH Client" running a computer with a Mosaic browser	The second computer system is shown in Fig. 1 as the HTTP Gateway and everything to its right	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
23	The web server (NCSA httpd) is shown as using the "CGI protocol" to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. "To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	"The prototype generates HTML documents on the fly from information pieces in the database." p. 1	The dynamic web page includes data dynamically retrieved from the "LINCKS Database Server." p. 6	LINCKS is a networked system	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks
24	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	The HTTP Gateway "parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." MAC004527	The InfoHarness Server obtains data from relational databases and "IHOs," or InfoHarness objects. "IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.	InfoHarness is networked. It "supports the largely automatic generation of InfoHarness repositories, and provides access to the original information from Mosaic and other World-Wide Web (WWW) browsers through an HTTP gateway." p. 2.	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
23	The web server (NCSA httpd) is shown as using the "CGI protocol" to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. "To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks	The dispatcher is that portion of cgi2lincks which decides where to route the request	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing.	"The prototype generates HTML documents on the fly from information pieces in the database." p. 1		'335	The references teaches a WWW front end to "LINCKS," an Object-Oriented Database Management System (OODBMS). "The prototype generates HTML documents on the fly from information pieces in the database." p. 1. "LINCKS fits very well in as a document or image server on a single site, and in particular the support of multiple views generated from the same information." Sec. 5	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks
24	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	"The InfoHarness server processes requests…" and the system provides "support for multiple clients" where the InfoHarness server processes the request while the HTTP Gateway is free to process other requests at the same time. MAC004527-28	The HTTP Gateway "parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." MAC004527	'335	The reference describes a system for generating dynamic web pages by passing requests from an HTTP Gateway (web server) to an InfoHarness Server. The InfoHarness Server generates "a combination of HTML forms and hyperlinks" and passes it back to the browser. pp. 4-5	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
23	The web server (NCSA httpd) is shown as using the "CGI protocol" to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. "To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing.	"The prototype generates HTML documents on the fly from information pieces in the database." p. 1	The dynamic web page includes data dynamically retrieved from the "LINCKS Database Server." p. 6	The references teaches a WWW front end to "LINCKS," an Object-Oriented Database Management System (OODBMS). "The prototype generates HTML documents on the fly from information pieces in the database." p. 1. "LINCKS fits very well in as a document or image server on a single site, and in particular the support of multiple views generated from the same information." Sec. 5	The dispatcher is that portion of cgi2lincks which decides where to route the request
24	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	"The InfoHarness server processes requests…" and the system provides "support for multiple clients" where the InfoHarness server processes the request while the HTTP Gateway is free to process other requests at the same time. MAC004527-28	The HTTP Gateway "parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." MAC004527	The InfoHarness Server obtains data from relational databases and "IHOs," or InfoHarness objects. "IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.	The reference describes a system for generating dynamic web pages by passing requests from an HTTP Gateway (web server) to an InfoHarness Server. The InfoHarness Server generates "a combination of HTML forms and hyperlinks" and passes it back to the browser. pp. 4-5	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
23	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	Cgi2lincks uses a key file to identify the appropriate w3lincks server, which accesses data sources	See '554 1d2		"When starting w3lincks, it connects to a LINCKS database and login as a normal user." p. 6		See '554 7	The tags in Sec. 4.3 are replaced with dynamically retrieved data. "For each of the nodes of the reference structure tree, we execute the statements in FORMATX part. Whenever the conditions in the FORMATX part is true, we execute the corresponding action which could result in the output of HTML or the extraction of images/sounds from the database." Sec. 4.2	The purpose of FORMATX is to express formatting
24	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	"IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The index is used to make the identification.	After the InfoHarness server retrieves the data, the HTTP Gateway "parses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." p.7				The Gateway "[p]arses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." p. 5. The conversion to HTML forms and hyperlinks is performed using tag-based (HTML) templates	The "response" of '554 8 is the dynamically retrieved data, which is inserted into the template	The purpose of the HTML template is precisely to drive the format of the data

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
23	FORMATX is a language for expressing HTML templates		LINCKS handles "composite objects" and "multiple views." Sec. 3.2. To handle objects it must process an object handling extension		LINCKS is computer-implemented	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks	The web server (NCSA httpd) is shown as using the “CGI protocol” to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. “To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	The request is intercepted because it is not handled by the Web server, but passed off to cgi2lincks	The Web server is the NCSA httpd server of Fig. 6. The page server, which generates the HTML page, is the combination of that portion of gci2lincks which adds HTML to a partially generated page and the w3lincks blocks in Fig. 6. The remainder of gci2lincks is part of the web server
24	See '554 8.		"The InfoHarness system architecture provides a platform for integrating information in a distributed environment by encapsulating existing and new information in objects, without converting, restructuring or reformatting the information. Through this object-oriented encapsulation, the system provides an integrated view and access to diverse and heterogeneous information." Sec. 2.0. The handling of objects is through an extension		InfoHarness is computer-implemented	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	The HTTP Gateway (web server) does not process the request, which instead is intercepted (not processed) and passed on to a dispatcher. See Fig. 1	The Web server is the HTTP Gateway. The page server is an InfoHarness Server. Routing is shown in Fig. 1

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
23	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing.	"The prototype generates HTML documents on the fly from information pieces in the database." p. 1	The dynamic web page includes data dynamically retrieved from the "LINCKS Database Server." p. 6	LINCKS is computer-implemented	The dispatcher is that portion of cgi2lincks which decides where to route the request	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	Cgi2lincks uses a key file to identify the appropriate w3lincks server, which accesses data sources	See '554 1d2	
24	"The InfoHarness server processes requests…" and the system provides "support for multiple clients" where the InfoHarness server processes the request while the HTTP Gateway is free to process other requests at the same time. MAC004527-28	The HTTP Gateway "parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." MAC004527	The InfoHarness Server obtains data from relational databases and "IHOs," or InfoHarness objects. "IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.	InfoHarness is computer-implemented	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	"IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness." p. 7. The index is used to make the identification.	After the InfoHarness server retrieves the data, the HTTP Gateway "parses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." p.7	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
23	"When starting w3lincks, it connects to a LINCKS database and login as a normal user." p. 6		See '554 7	The tags in Sec. 4.3 are replaced with dynamically retrieved data. "For each of the nodes of the reference structure tree, we execute the statements in FORMATX part. Whenever the conditions in the FORMATX part is true, we execute the corresponding action which could result in the output of HTML or the extraction of images/sounds from the database." Sec. 4.2	The purpose of FORMATX is to express formatting	FORMATX is a language for expressing HTML templates		LINCKS handles "composite objects" and "multiple views." Sec. 3.2. To handle objects it must process an object handling extension		LINCKS is computer-implemented
24			The Gateway "[p]arses the response, converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser." p. 5. The conversion to HTML forms and hyperlinks is performed using tag-based (HTML) templates	The "response" of '554 8 is the dynamically retrieved data, which is inserted into the template	The purpose of the HTML template is precisely to drive the format of the data	See '554 8.		"The InfoHarness system architecture provides a platform for integrating information in a distributed environment by encapsulating existing and new information in objects, without converting, restructuring or reformatting the information. Through this object-oriented encapsulation, the system provides an integrated view and access to diverse and heterogeneous information." Sec. 2.0. The handling of objects is through an extension		InfoHarness is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
23	The dispatcher is that portion of cgi2lincks which decides where to route the request	“LINCKS supports both dynamically bound links and statically bound links. A dynamically bound link is between a source object and a target object where the binding to a particular version of the target object occurs on demand.” Sec. 2.3. The links are maintained in General Presentation Descriptors (GPDs)	Cgi2links reads a key file which identifies the server on which w3lincks is running, thus making a selection of page server informed by how to connect to it. "When starting w3lincks, it connects to a LINCKS database and login as a normal user It writes the key file file and waits for a connection from cgi2lincks." Sec 4.2. Thus the key file maintains dynamic information abut the page server	Fig. 6 shows a request being routed from the web server (NCSA httpd) through cgi2lincks to w3lincks	The web server (NCSA httpd) is shown as using the “CGI protocol” to send the request to cgi2lincks. If the page server is on the same machines as the web server, the interprocess mechanism for invoking CGI involves SBL releasing, according to EpicRealm's contentions. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. “To enhance the performance of a system for processing dynamic web page generation requests, it would have been obvious for a person of ordinary skill to utilize a page server on a machine separate from the web server." MAC000904. Further, to a person of ordinary skill, it would have been obvious to try moving the cgi2lincks to the page server machine. Methods for network communications were well-known and therefore, communicating the request from the NCSA httpd server to the page server machine hosting cgi2lincks and w3lincks would have been accomplished using TCP/IP, implying TCP releasing.	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. Section 4.1 teaches running w3lincks on a different host (machine) from the one on which cgi2lincks runs. Therefore, it must be on a different machine from the web server and the processing being performed by w3lincks cannot affect what the web server (NCSA httpd) is doing.	“The prototype generates HTML documents on the fly from information pieces in the database.” p. 1	The dynamic web page includes data dynamically retrieved from the “LINCKS Database Server.” p. 6
24	Fig. 1 shows multiple InfoHarness Servers, each of which has access to potentially different data. Therefore, the Gateway must make an informed selection which InfoHarness Server will receive the request, and thus performs the function of dispatcher	"The Repository Generator that is used for off-line automatic generation of meta-data that represents the desirable view on the structure and organization of the original information. This meta-data is used at run-time by the InfoHarness server to search and retrieve information." p. 4. The information generated by the Repository Generator is dynamic, i.e. constantly changing	The Gateway passes the request to the selected InfoHarness server. See Fig. 1. See also '554 1b5. "The autonomy of administrators, no centralized control, support for multiple clients and multiple servers in a large-scale distributed, heterogeneous environment all support the goals of scalability and operability in a large and geographically distributed environment." Sec. 2.1. "Scalability" implies informed selection of page servers based on dynamic information	The page server cannot process the request unless it receives it. Receiving is shown in Fig. 1.	The HTTP Gateway passes the request using HTTP, so TCP releasing occurs according to EpicRealm's contention. The Web server establishes a socket connection with the InfoHarness Server. MAC004527. This provides SBL releasing, according to EpicRealm's contention	“The InfoHarness server processes requests…” and the system provides “support for multiple clients” where the InfoHarness server processes the request while the HTTP Gateway is free to process other requests at the same time. MAC004527-28	The HTTP Gateway “parses the response [received from the InfoHarness server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser.” MAC004527	The InfoHarness Server obtains data from relational databases and “IHOs,” or InfoHarness objects. “IHOs that encapsulate indexed collections store information about the location of both the index and the query method. Any indexed collection may make use of its own data retrieval method that is not part of InfoHarness.” p. 7. The Gateway then "[p]arses the response [from the InfoHarness Server], converts it to a combination of HTML forms and hyperlinks, adds the HTTP header, and passes the transformed response to the Mosaic browser," thus creating the dynamic page. p. 5.

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
25	058	Hastings and Kumar, Providing Customers Information Using the Web and CORBA	Oct. 1994		'554	The reference as a whole teaches producing Web pages in response to time-varying transactional data by allowing information from databases to be included in web pages constructed on the fly. See also p. 4	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The Web Broker receives the request in Fig. 2.	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker
26	153	Oracle WebServer User's Guide Release 1.0 Part Number A34986-2	1995		'554	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." MAC003052	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
25	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The HTTPD (web) server processes other requests as the Application is retrieving data.	A web page is dynamically generated in response to the request, as shown in Fig. 3	The page includes "data from production systems," i.e. dynamically retrieved data.	The required identification is performed by the Object Broker. "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5	The Web Broker moves the user's data from the HTML form and creates an ObjectBroker object and invokes the remote method to service the user's request (obtain data dynamically) p. 4.
26	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates asynchronously. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Also, OWS 1.0 allows for instantiation of multiple web listeners, allowing for concurrent processing of requests.	See '554 1c	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.	"After connecting to the database the Web Agent invokes the appropriate PL/SQL procedure, whose name is obtained by parsing the PATH_INFO environment variable." MAC003146. The data sources to be used are identified by examining the PATH_INFO environment variable.	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 Server that enables the creation of dynamic HTML documents from data stored in an Oracle database. When the data changes, these HTML documents are updated automatically….This approach supplements the presentation of static, or unchanging, data which is found on most sites today, with the dynamic real-time data present in business systems based on the Oracle 7 Server." p. 1-2 (MAC003052). 5. The PL/SQL procedure executes, generating an HTML document. With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." (Oracle, p. 5-5).

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
25		The user is required to provide authorization (log in) to obtain data: "A key requirement of the system was to provide a per-service based access control mechanism. For the current version, access control is implemented using the authentication scheme provided by the NCSA's HTTPD server. This scheme requires maintaining access control files in specific directories of the HTTPD server directory structure. The User Authorization and Profile Subsystem allows an Administrator to perform this function." p. 7		"The Meta HTML files reside on the Web Server system. These files contain supersets of HTML information to substitute data returned from the production systems, handle conditional operations based on that data, and implement location specific date, time, and monetary functions." p.5. "Supersets" are extensions.	See '554 7.	The system of the reference is Web-based, hence networked	The page includes "data from production systems," i.e. dynamically retrieved data.	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.
26	Connection caching is used in SQL*Net, which may optionally be used with OWS 1.0. ORCL01806717	"In order to connect to an Oracle 7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (MAC003146). The username and password are used to log in to the data source. "A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." (p. 1-4 (MAC 3054; ORCL 23). "When the Oracle Web Agent logs into the Oracle7 Server, it starts a PL/SQL procedure that has been created by the user to generate an HTML page as its output." (p. 1-4 (MAC 3054; ORCL 23).)	Pages MAC003097-98, MAC003223-24 and MAC003255-56 explain how to set up file caching, which "decreases the access time required for commonly-used files." "The Oracle Web Listener allows a configurable set of commonly accessed files to be cached in memory. This provides very good performance when these files are accessed by a client." MAC003086	An extension template is a construct that is not HTML but that can easily be converted into HTML. It "extends" HTML by providing the user with additional facilities. Oracle Webserver provides a Developer's Toolkit incorporating hypertext procedures (HTP), described in Section 6: "A hypertext procedure generates a line in an HTML document that contains the HTML tag that corresponds to its name. For instance, the htp.anchor procedure generates an anchor tag." A variety of hypertext procedures are provided with the toolkit, including "Body Tags," "List Tags," "Form Tags" and "Table Tags" which "allow the user to insert tables and manipulate the size and columns of the table in a document. All of these are HTML extension templates	The reference gives many examples of inserting dynamic data obtained by accessing an SQL database into an HTML extension template. For example, the procedure htp.cite, described on page 6-20 of the reference (MAC003182), when called with two arguments, e.g. htp.cite(ctext, cattributes), causes the arguments to be substituted into a template as follows: <CITE cattributes>ctext</CITE>. See PL/SQL procedures at MAC003155-3157	OIS operates with a Web server, hence is networked	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053. "The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. A processor of a computer that runs Oracle7 server is a page server processing means.

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
25	This means is the "Customer Using PC, Workstation, Mac, or Terminal," as shown in Fig. 1	The second computer system is the WSS System of Fig. 1	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The Web Broker receives the request in Fig. 2.
26	The first computer system is a computer that has a processor that runs a web browser as shown on page 1-3 (MAC003053).	The second computer system is a computer having a processor that runs the Web Listener is shown in a rectangular box on p. 1-3, MAC003053.	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053. The means is a processor of a computer that runs the Web Listener. See figure on MAC003053.	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
25	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	A web page is dynamically generated in response to the request, as shown in Fig. 3	The page includes "data from production systems," i.e. dynamically retrieved data.	The system of the reference is Web-based, hence networked	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	Because the system is computerized, its instructions must be on a machine-readable medium.	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The Web Broker receives the request in Fig. 2.
26	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	See '554 1c	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.	OIS operates with a Web server, hence is networked	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	OIS is a programmed system whose instructions are on a machine readable medium	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
25	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The HTTPD (web) server processes other requests as the Application is retrieving data.	A web page is dynamically generated in response to the request, as shown in Fig. 3	'335	The reference as a whole teaches producing Web pages in response to time-varying transactional data by allowing information from databases to be included in web pages constructed on the fly. See also p. 4	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The Web Broker receives the request in Fig. 2.
26	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates asynchronously. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Also, OWS 1.0 allows for instantiation of multiple web listeners, allowing for concurrent processing of requests.	See '554 1c	'335	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." MAC003052	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
25	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The HTTPD (web) server processes other requests as the Application is retrieving data.	A web page is dynamically generated in response to the request, as shown in Fig. 3	The page includes "data from production systems," i.e. dynamically retrieved data.	The reference as a whole teaches producing Web pages in response to time-varying transactional data by allowing information from databases to be included in web pages constructed on the fly. See also p. 4	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request
26	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates asynchronously. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Also, OWS 1.0 allows for instantiation of multiple web listeners, allowing for concurrent processing of requests.	See '554 1c	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." MAC003052	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
25	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The required identification is performed by the Object Broker. "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5	The Web Broker moves the user's data from the HTML form and creates an ObjectBroker object and invokes the remote method to service the user's request (obtain data dynamically) p. 4.		The user is required to provide authorization (log in) to obtain data: "A key requirement of the system was to provide a per-service based access control mechanism. For the current version, access control is implemented using the authentication scheme provided by the NCSA's HTTPD server. This scheme requires maintaining access control files in specific directories of the HTTPD server directory structure. The User Authorization and Profile Subsystem allows an Administrator to perform this function." p. 7		The Meta HTML files are tag-based text templates.	See '554 7.	"The OWL [Object Wrapper Layer] remote method returns multiple lines, string arrays, of tag/value pairs that contain the results. The tags in the returned data correspond to the META HTML tag names. These are substituted according to the following rules to create the HTML output. This process gives a completely mapping free implementation of the data formats, only the Object Wrapper Server and META HTML routines need to know the tags names." p. 6
26	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	"After connecting to the database the Web Agent invokes the appropriate PL/SQL procedure, whose name is obtained by parsing the PATH_INFO environment variable." MAC003146. The data sources to be used are identified by examining the PATH_INFO environment variable.	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 Server that enables the creation of dynamic HTML documents from data stored in an Oracle database. When the data changes, these HTML documents are updated automatically….This approach supplements the presentation of static, or unchanging, data which is found on most sites today, with the dynamic real-time data present in business systems based on the Oracle 7 Server." p. 1-2 (MAC003052). 5. The PL/SQL procedure executes, generating an HTML document. With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." (Oracle, p. 5-5).	Connection caching is used in SQL*Net, which may optionally be used with OWS 1.0. ORCL01806717	"In order to connect to an Oracle 7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (MAC003146). The username and password are used to log in to the data source. "A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." (p. 1-4 (MAC 3054; ORCL 23). "When the Oracle Web Agent logs into the Oracle7 Server, it starts a PL/SQL procedure that has been created by the user to generate an HTML page as its output." (p. 1-4 (MAC 3054; ORCL 23).)	Pages MAC003097-98, MAC003223-24 and MAC003255-56 explain how to set up file caching, which "decreases the access time required for commonly-used files." "The Oracle Web Listener allows a configurable set of commonly accessed files to be cached in memory. This provides very good performance when these files are accessed by a client." MAC003086	OIS uses HTML extension templates, which are tag-based. See '554 7	See '554 8.	Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
25	Meta HTML files without extensions are HTML templates.		CORBA is object-based. It is accessed through an object handling extension		The system of the reference is computer-implemented	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.	The Web Broker receives the request in Fig. 2.	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Web Broker	The Web server (HTTPD – Mosaic and Lynx Server in Fig. 2) routes requests to the Web Broker, which in turn sends them to the "Existing Production Application." The page server is the combination of the Web Broker and the Existing Production Application because Fig. 3 shows the web page being produced at the Web Broker by replacing HTML tags with data obtained from the Application.
26	See '554 8.	Dynamically updating the database is described on page 4-9 (MAC003113). "Which SQL statements determine the actual data content? Chiefly, three – INSERT, UPDATE, and DELETE. INSERT places rows in a table, UPDATE changes the values they contain, and DELETE removes them." p. 4-8 (MAC003112)			OIS is computer-implemented	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	"When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." MAC003053. Thus the Web Listener performs interception	"The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." MAC003052. Thus the Oracle7 Server is a "page server" of the patent. The dynamic page generation request is received by the web listener (web server) and routed to the Oracle7 server, as shown in the diagram on page MAC003053.

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
25	The HTTPD (web) server processes other requests as the Application is retrieving data.	A web page is dynamically generated in response to the request, as shown in Fig. 3	The page includes "data from production systems," i.e. dynamically retrieved data.	The system of the reference is computer-implemented	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The required identification is performed by the Object Broker. "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5	The Web Broker moves the user's data from the HTML form and creates an ObjectBroker object and invokes the remote method to service the user's request (obtain data dynamically) p. 4.	
26	The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates asynchronously. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Also, OWS 1.0 allows for instantiation of multiple web listeners, allowing for concurrent processing of requests.	See '554 1c	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.	OIS is computer-implemented	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	"After connecting to the database the Web Agent invokes the appropriate PL/SQL procedure, whose name is obtained by parsing the PATH_INFO environment variable." MAC003146. The data sources to be used are identified by examining the PATH_INFO environment variable.	"The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 Server that enables the creation of dynamic HTML documents from data stored in an Oracle database. When the data changes, these HTML documents are updated automatically….This approach supplements the presentation of static, or unchanging, data which is found on most sites today, with the dynamic real-time data present in business systems based on the Oracle 7 Server." p. 1-2 (MAC003052). 5. The PL/SQL procedure executes, generating an HTML document. With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." (Oracle, p. 5-5).	Connection caching is used in SQL*Net, which may optionally be used with OWS 1.0. ORCL01806717

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
25	The user is required to provide authorization (log in) to obtain data: "A key requirement of the system was to provide a per-service based access control mechanism. For the current version, access control is implemented using the authentication scheme provided by the NCSA's HTTPD server. This scheme requires maintaining access control files in specific directories of the HTTPD server directory structure. The User Authorization and Profile Subsystem allows an Administrator to perform this function." p. 7		The Meta HTML files are tag-based text templates.	See '554 7.	"The OWL [Object Wrapper Layer] remote method returns multiple lines, string arrays, of tag/value pairs that contain the results. The tags in the returned data correspond to the META HTML tag names. These are substituted according to the following rules to create the HTML output. This process gives a completely mapping free implementation of the data formats, only the Object Wrapper Server and META HTML routines need to know the tags names." p. 6	Meta HTML files without extensions are HTML templates.		CORBA is object-based. It is accessed through an object handling extension		The system of the reference is computer-implemented
26	"In order to connect to an Oracle 7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." (MAC003146). The username and password are used to log in to the data source. "A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." (p. 1-4 (MAC 3054; ORCL 23). "When the Oracle Web Agent logs into the Oracle7 Server, it starts a PL/SQL procedure that has been created by the user to generate an HTML page as its output." (p. 1-4 (MAC 3054; ORCL 23).)	Pages MAC003097-98, MAC003223-24 and MAC003255-56 explain how to set up file caching, which "decreases the access time required for commonly-used files." "The Oracle Web Listener allows a configurable set of commonly accessed files to be cached in memory. This provides very good performance when these files are accessed by a client." MAC003086	OIS uses HTML extension templates, which are tag-based. See '554 7	See '554 8.	Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page	See '554 8.	Dynamically updating the database is described on page 4-9 (MAC003113). "Which SQL statements determine the actual data content? Chiefly, three – INSERT, UPDATE, and DELETE. INSERT places rows in a table, UPDATE changes the values they contain, and DELETE removes them." p. 4-8 (MAC003112)			OIS is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
25	The Web Broker serves as a dispatcher, which learns through the Object Broker which production application holds the data necessary to fulfill the request	See '554 1b5	The Web Broker makes an informed selection of the application (page server) to which it forwards the request. The function of the Web Broker is to "select and connect to a production system based on user's request type, country and authorization." Fig. 2. The selection is done by dynamic invocation of the Object Broker, which maintains dynamic information about page servers: "Here is one of the strengths of using the CORBA based system. The application code does not have to know anything about the remote node being identified or about the mechanics of the transport being used. That is invisible to the application being supplied and managed by ObjectBroker." p. 5.	The Web Broker receives the request in Fig. 2.	The Object Broker integrates with HTTP, which makes use of TCP, so there is TCP releasing according to EpicRealm's contentions.	The HTTPD (web) server processes other requests as the Application is retrieving data.	A web page is dynamically generated in response to the request, as shown in Fig. 3	The page includes "data from production systems," i.e. dynamically retrieved data.
26	The web agent in OracleServer acts as a dispatcher. "The Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer." MAC003054. Thus the Web Agent makes an informed choice of which Web Agent service is to be invoked to handle the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." MAC003148. This is configuration file dispatching.	Page 4-5 (MAC003109) describes how the page server (Oracle7 Server) can access multiple databases and teaches a facility known as SQL*Net, which communicates with components of a distributed database. It is inherent in this structure that the page server maintains information on which data sources it may access. "If you have a full license for the Oracle7 Server, you can use it to create distributed databases. These are multiple Oracle7 databases – located on different machines and possibly separated by considerable geographical distance – that work together to maintain a single consistent body of data that looks and behaves as though drawn from a single source. In this situation, you connect to one database, which is called the local database, and that database interacts with other databases, called remote databases, as needed." p. 4-5 (MAC003109)	Oracle WebServer 1.0 also performs SQL*Net dispatching. Page 1-3. "To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly." MAC003148-3150.	The diagram on page MAC003053 shows that the request is received by the Oracle7 Server (page server).	"For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." p. 3-2. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the web listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. Further, OIS passes requests using HTTP, so TCP releasing is performed. Page 1-3 shows SQL*Net running over a TCP/IP connection. The "releasing limitation" is met under the TCP theory of releasing. Furthermore, the Oracle7 server binds to a TCP port and therefore the releasing limitation is met under the SBL theory of releasing. MAC003053	The Oracle7 server (page server) processes the dynamic page generation request and returns the generated page to the web agent. This process occurs simultaneously with the web listener (web server) processing other requests because the Web Listener operates asynchronously. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." MAC003053. Also, OWS 1.0 allows for instantiation of multiple web listeners, allowing for concurrent processing of requests.	See '554 1c	The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database. "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." MAC003054.

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
27	141	Derler, "The World-Wide Web Gateway to Hyper-G: Using a Connectionless Protocol to Access Session-Oriented Services"	Mar. 1995		'554	"When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4.) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items." Sec. 3.3.4	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2
28	051	Bowman, Harvest: A Scalable, Customizable Discovery and Access System	Mar. 12, 1995		'554	Dynamic pages are summaries generated from documents. "generating content summaries that range in detail from full text to highly abridged and specialized signatures" p. 9. Dynamic pages are also formatted result sets from queries (requests)	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	The request must be received to be processed	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	Intercepting is recognizing a "form submission or other database query" p. 2

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
27	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2	The reference discloses processing by the HyperG server while the web server processes other requests at the same time in its discussion of parallel processing: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13	See '554 1d1. The "data sources" are the databases	The Hyper-G server, in processing the request, identifies one or more data sources from which to retrieve data. Sec. 3.3.4. "The Hyper-G server is able to store 'remote objects' (i.e. Hyper-G objects in the link database that are pointers to remote systems) for accessing WWW, Gopher and WAIS servers. This allows Hyper-G clients to contact such remote systems and presenting a 'Hyper-G view' (as far as possible) of information on non-Hyper-G servers."	During the process where the Hyper-G system dynamically generates a web page, the Hyper-G server dynamically retrieves data from one or more data sources, i.e., the Hyper-G server's database or other data source. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"); Figure 4.13. Also Section 3.3.4
28	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	Concurrent processing is inherent because distinct machines are used	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	This occurs when the appropriate Broker is identified. "Efficiency and flexibility are important because we want to enable the construction of many different topic specific Brokers. For example, one Broker might index scientific papers for microbiologists while another Broker indexes PC software archives ... Harvest includes a distinguished Broker called the Harvest Server Registry HSR, which allows users to register information about each Harvest Gatherer, Broker Cache, and Replicator in the Internet. The HSR is useful when searching for an appropriate Broker and when constructing new Gatherers and Brokers, to avoid duplication of effort." p. 6	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
27	"In contrast to the concept of handling every single WWW request as a separate Hyper-G session, the implementation of the new gateway described in this chapter is based on the idea of keeping up the connection between the gateway and the Hyper-G server during the processing of consecutive HTTP requests." Sec. 4.2.1. This maintaining of the connection constitutes a connection cache.	"In this example, uname and secretpassw are the user's Hyper-G user name and password. If they correspond to a valid account on the Hyper-G server, the user will be logged in as an identified user with probably extended access rights and the gateway responds by sending the file ident_ok.html." Sec. 4.6.2	Page caching is an important aspect of Hyper-G. "[I]n the Hyper-G system, requests from clients rather than being directly sent to document servers, are routed through a document cache. … If the document requested has previously been cached and can be found in the cache server, it is transmitted directly. Otherwise, the cache server retrieves the document from the document server, transmits it to the client and stores a copy of it in the cache." Sec. 3.3.5	HTML extensions are described in Sec. 4.6.1.1, where non-HTML "place holders" are added to documents	"The following place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1	The reference teaches a networked system for managin dynamic page generation requests.	See '554 1d1. The "data sources" are the databases	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3
28	"Because this poses so much more load on Provider-site servers we implemented a connection caching mechanism which attempts to keep connections open across individual object retrievals." p. 10		"The Harvest Object Cache reduces network load, server load, and response latency when accessing located information objects." p. 6. "For example Mosaic caches recently accessed documents in virtual memory. Previous queries can be retrieved and refined by moving through the cache" p. 17			Harvest is a networked system: ""Harvest reduces network traffic four ways: extracting indexing data before transmitting across the network, supporting incremental updates, transmitting the data in compressed form from Gatherers to Brokers and by using a local cache of recently retrieved objects." p. 10	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
27	The first computer system includes the client browser	The second computer system includes the WWW gateway	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1
28	The means is the client computer. Fig. 4, p. 8	The second computer system is Fig. 4 minus the client computer	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	Intercepting is recognizing a "form submission or other database query" p. 2	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
27	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13	See '554 1d1. The "data sources" are the databases	The reference teaches a networked system for managin dynamic page generation requests.	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2	Hyper-G is a programmed system whose instructions are on a machine readable medium	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1
28	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	Harvest is a networked system: ""Harvest reduces network traffic four ways: extracting indexing data before transmitting across the network, supporting incremental updates, transmitting the data in compressed form from Gatherers to Brokers and by using a local cache of recently retrieved objects." p. 10	Intercepting is recognizing a "form submission or other database query" p. 2	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	Because Harvest is a programmed system, it must have a medium on which instructions are stored.	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
27	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2	The reference discloses processing by the HyperG server while the web server processes other requests at the same time in its discussion of parallel processing: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13	'335	"When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4.) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items." Sec. 3.3.4	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1
28	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	Intercepting is recognizing a "form submission or other database query" p. 2	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	Concurrent processing is inherent because distinct machines are used	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	'335	Dynamic pages are summaries generated from documents. "generating content summaries that range in detail from full text to highly abridged and specialized signatures" p. 9. Dynamic pages are also formatted result sets from queries (requests)	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
27	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	The reference discloses processing by the HyperG server while the web server processes other requests at the same time in its discussion of parallel processing: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing “integration of WWW databases into Hyper-G”). Such a page is shown in Fig. 4.13	See '554 1d1. The "data sources" are the databases	“When a WWW client requests a Hyper-G collection, the WWW gateway (see chapter 4.) will generate an HTML document containing a list of documents (including sub-collections) belonging to the collection with hypertext links to all these items.” Sec. 3.3.4	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G’s search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.
28	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	Intercepting is recognizing a "form submission or other database query" p. 2	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	Concurrent processing is inherent because distinct machines are used	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	Dynamic pages are summaries generated from documents. "generating content summaries that range in detail from full text to highly abridged and specialized signatures" p. 9. Dynamic pages are also formatted result sets from queries (requests)	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
27	The selection is based on dynamic information maintained about page servers. “The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc.” Sec. 4.2.1. Dispatching is also based in the reference on server load: “The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user.” Sec. 4.2.2	The Hyper-G server, in processing the request, identifies one or more data sources from which to retrieve data. Sec. 3.3.4. "The Hyper-G server is able to store ‘remote objects’ (i.e. Hyper-G objects in the link database that are pointers to remote systems) for accessing WWW, Gopher and WAIS servers. This allows Hyper-G clients to contact such remote systems and presenting a ‘Hyper-G view’ (as far as possible) of information on non-Hyper-G servers.”	During the process where the Hyper-G system dynamically generates a web page, the Hyper-G server dynamically retrieves data from one or more data sources, i.e., the Hyper-G server’s database or other data source. Sec. 4.1 (referencing “integration of WWW databases into Hyper-G”); Figure 4.13. Also Section 3.3.4	"In contrast to the concept of handling every single WWW request as a separate Hyper-G session, the implementation of the new gateway described in this chapter is based on the idea of keeping up the connection between the gateway and the Hyper-G server during the processing of consecutive HTTP requests." Sec. 4.2.1. This maintaining of the connection constitutes a connection cache.	"In this example, uname and secretpassw are the user's Hyper-G user name and password. If they correspond to a valid account on the Hyper-G server, the user will be logged in as an identified user with probably extended access rights and the gateway responds by sending the file ident_ok.html." Sec. 4.6.2	Page caching is an important aspect of Hyper-G. “[I]n the Hyper-G system, requests from clients rather than being directly sent to document servers, are routed through a document cache. … If the document requested has previously been cached and can be found in the cache server, it is transmitted directly. Otherwise, the cache server retrieves the document from the document server, transmits it to the client and stores a copy of it in the cache.” Sec. 3.3.5	Tag-based text templates are described in Sec. 4.6.1.1	"The following place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1	The purpose of the templates is to format the data
28	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	This occurs when the appropriate Broker is identified. "Efficiency and flexibility are important because we want to enable the construction of many different topic specific Brokers. For example, one Broker might index scientific papers for microbiologists while another Broker indexes PC software archives ... Harvest includes a distinguished Broker called the Harvest Server Registry HSR, which allows users to register information about each Harvest Gatherer, Broker Cache, and Replicator in the Internet. The HSR is useful when searching for an appropriate Broker and when constructing new Gatherers and Brokers, to avoid duplication of effort." p. 6	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	"Because this poses so much more load on Provider-site servers we implemented a connection caching mechanism which attempts to keep connections open across individual object retrievals." p. 10		"The Harvest Object Cache reduces network load, server load, and response latency when accessing located information objects." p. 6. "For example Mosaic caches recently accessed documents in virtual memory. Previous queries can be retrieved and refined by moving through the cache" p. 17	"Each template contains a type, a Uniform Resource Locator (URL), and a list of byte count delimited attribute value pairs." p. 6. (the attributes are the tags.) Also Fig. 3. "We built summarizers to translate these descriptions from the formats used by six different archive sites into uniformly structured SOIF templates. This Broker demonstrates Harvest's ability to take advantage of high quality manual annotation information when it is available ..." p. 23	Dynamic retrieval is done by Glimpse. "one of our search engines (Glimpse) returns the matching SOIF attribute for each summary object in the result list. The Query Manager constructs a short object description from the OID the search engine specific data and the summary object It then returns the list of object descriptions to the client" p. 13	Depends on construction of "format", but this is what the templates do. "Our PC software Broker demonstrates Harvest's ability to incorporate manually generated indexing information in a variety of formats … We built summarizers to translate these descriptions from the formats used by six different archive sites into uniformly structured SOIF templates." p. 22

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
27	The templates are HTML-based. See Sec. 4.6.1.2		Hyper-G maintains objects, and has special methods for handling the: "This link server is an object oriented database. It holds information about Hyper-G "objects", which can be links, anchors, descriptions of documents, relations between such objects (e.g. which documents belong to which collection, which source anchors are attached to which destination anchors, which anchors are attached to which documents, etc.) and additional attributes such as title, author, creation date, modification date, etc." Sec. 3.2.3		Hyper-G is computer-implemented	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The Master Process of the Web server intercepts the handling of the request; the Master Process does not process the request but spawns a slave process to handle it. Sec. 4.2.2	The Web server is the WWW gateway. The page server is the Hyper-G server. "When it has accepted a connection initiated by a passing process, it reads the request, processes it (usually by transforming it into a Hyper-G request for the server)" Sec. 4.2.3
28	"The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker's database and to the original resource described by the summary object." p. 15	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes. p. 5	Harvest handles object through extensions: "Gatherers and Brokers communicate using an attribute/value stream protocol called the Summary Object Interchange Format (SOIF) [27] intended to be easily parsed yet sufficiently expressive to handle many kinds of objects." Sec. 3		Harvest is a computer system	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15	The request must be received to be processed	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	Intercepting is recognizing a "form submission or other database query" p. 2	Web server is the HTTP server, p. 8. Page server is the Broker. "The gateway retrieves the query from the form and sends it to the query manager. The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker s database and to the original resource described by the summary object" p. 15

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
27	The reference discloses processing by the HyperG server while the web server processes other requests at the same time in its discussion of parallel processing: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13	See '554 1d1. The "data sources" are the databases	Hyper-G is computer-implemented	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2	The Hyper-G server, in processing the request, identifies one or more data sources from which to retrieve data. Sec. 3.3.4. "The Hyper-G server is able to store 'remote objects' (i.e. Hyper-G objects in the link database that are pointers to remote systems) for accessing WWW, Gopher and WAIS servers. This allows Hyper-G clients to contact such remote systems and presenting a 'Hyper-G view' (as far as possible) of information on non-Hyper-G servers."	During the process where the Hyper-G system dynamically generates a web page, the Hyper-G server dynamically retrieves data from one or more data sources, i.e., the Hyper-G server's database or other data source. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"); Figure 4.13. Also Section 3.3.4	"In contrast to the concept of handling every single WWW request as a separate Hyper-G session, the implementation of the new gateway described in this chapter is based on the idea of keeping up the connection between the gateway and the Hyper-G server during the processing of consecutive HTTP requests." Sec. 4.2.1. This maintaining of the connection constitutes a connection cache.
28	Concurrent processing is inherent because distinct machines are used	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	Harvest is a computer system	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	This occurs when the appropriate Broker is identified. "Efficiency and flexibility are important because we want to enable the construction of many different topic specific Brokers. For example, one Broker might index scientific papers for microbiologists while another Broker indexes PC software archives ... Harvest includes a distinguished Broker called the Harvest Server Registry HSR, which allows users to register information about each Harvest Gatherer, Broker Cache, and Replicator in the Internet. The HSR is useful when searching for an appropriate Broker and when constructing new Gatherers and Brokers, to avoid duplication of effort." p. 6	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5	"Because this poses so much more load on Provider-site servers we implemented a connection caching mechanism which attempts to keep connections open across individual object retrievals." p. 10

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
27	"In this example, uname and secretpassw are the user's Hyper-G user name and password. If they correspond to a valid account on the Hyper-G server, the user will be logged in as an identified user with probably extended access rights and the gateway responds by sending the file ident_ok.html." Sec. 4.6.2	Page caching is an important aspect of Hyper-G. "[I]n the Hyper-G system, requests from clients rather than being directly sent to document servers, are routed through a document cache. … If the document requested has previously been cached and can be found in the cache server, it is transmitted directly. Otherwise, the cache server retrieves the document from the document server, transmits it to the client and stores a copy of it in the cache." Sec. 3.3.5	Tag-based text templates are described in Sec. 4.6.1.1	"The following place holders, if contained in HTML files of the gateway, are replaced with their current values before the contents of those files are sent to the client." Sec. 4.6.1.1	The purpose of the templates is to format the data	The templates are HTML-based. See Sec. 4.6.1.2		Hyper-G maintains objects, and has special methods for handling the: "This link server is an object oriented database. It holds information about Hyper-G "objects", which can be links, anchors, descriptions of documents, relations between such objects (e.g. which documents belong to which collection, which source anchors are attached to which destination anchors, which anchors are attached to which documents, etc.) and additional attributes such as title, author, creation date, modification date, etc." Sec. 3.2.3		Hyper-G is computer-implemented
28		"The Harvest Object Cache reduces network load, server load, and response latency when accessing located information objects." p. 6. "For example Mosaic caches recently accessed documents in virtual memory. Previous queries can be retrieved and refined by moving through the cache" p. 17	"Each template contains a type, a Uniform Resource Locator (URL), and a list of byte count delimited attribute value pairs." p. 6. (the attributes are the tags.) Also Fig. 3. "We built summarizers to translate these descriptions from the formats used by six different archive sites into uniformly structured SOIF templates. This Broker demonstrates Harvest's ability to take advantage of high quality manual annotation information when it is available ..." p. 23	Dynamic retrieval is done by Glimpse. "one of our search engines (Glimpse) returns the matching SOIF attribute for each summary object in the result list. The Query Manager constructs a short object description from the OID the search engine specific data and the summary object It then returns the list of object descriptions to the client" p. 13	Depends on construction of "format", but this is what the templates do. "Our PC software Broker demonstrates Harvest's ability to incorporate manually generated indexing information in a variety of formats … We built summarizers to translate these descriptions from the formats used by six different archive sites into uniformly structured SOIF templates." p. 22	"The set of objects returned by the Query Manager is converted into an HTML page with hypertext links to complete summary objects in the Broker's database and to the original resource described by the summary object." p. 15	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes. p. 5	Harvest handles object through extensions: "Gatherers and Brokers communicate using an attribute/value stream protocol called the Summary Object Interchange Format (SOIF) [27] intended to be easily parsed yet sufficiently expressive to handle many kinds of objects." Sec. 3		Harvest is a computer system

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
27	The master process passes the request to a slave process (dispatcher). Secs. 4.2.2, 4.2.3; Figs. 4.1, 4.2. The slave process (dispatcher) makes an informed selection about which Hyper-G server should receive the request, based on Hyper-G's search facility, which locates the server on which the required information object resides. Secs. 3.3.1, 3.5.	The Hyper-G system maintains dynamic information regarding data sources a given Hyper-G server may access. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1	The selection is based on dynamic information maintained about page servers. "The stored information can be for instance the object ID of the last retrieved object, the user name of identified users, information needed to distinguish requests from different users, parameters influencing the optical appearance of collection lists, etc. This information is used, for example, to determine the scope for searches, for user identification purposes, for keeping certain parameters which can be set by users, etc." Sec. 4.2.1. Dispatching is also based in the reference on server load: "The amount of time spent on processing may vary depending on network load, server load, etc. If the gateway is processing consecutive requests from several clients, some requests might have to wait for acceptance by the gateway for a long time. The problem is not so much the amount of time the client has to wait until the response has been received from the gateway since waiting a few seconds for the completion of processing in some rare cases is acceptable for the user." Sec. 4.2.2	Receiving is taught expressly: "This means that a connection between the gateway and the Hyper-G server is established when the gateway receives the first request of a new user (i.e. a user for whom currently no Hyper-G session is being maintained). This connection is being kept up during the processing of consecutive requests from the same user and is closed at the end of a session." Sec. 4.2.1	Figure 4.17 shows the request being passed from the WWW Gateway (web server) to the page server via HTTP 1.0. Derler teaches that TCP/IP is used for client-server connections, and Derler further discloses that the WWW Gateway is a client of the Hyper-G Server. See MAC002675, 2711. Thus TCP releasing occurs according to EpicRealm's contentions	The reference discloses processing by the HyperG server while the web server processes other requests at the same time in its discussion of parallel processing: "To avoid this problem, the task has to be split in such a way that the process which accepts connections from the "outside" is not burdened with the actual processing of the request, but instead is allowed to accept further requests while others are still being processed." Sec. 4.2.2	The Hyper-G system dynamically generates a Web page in response to the request. Sec. 4.1 (referencing "integration of WWW databases into Hyper-G"). Such a page is shown in Fig. 4.13	See '554 1d1. The "data sources" are the databases
28	Dispatching is determining which Broker (page server) to use to handle the query. In Harvest this is an informed selection based on dynamic information. "Harvest replicates its Brokers with the mirror-d replication tool … Basically, mirror-d dynamically configures the ftp-mirror software between replicas in a replication group so that updates propagate along the highest bandwidth, lowest delay network paths." p. 17. "Gatherers and Brokers can be arranged in various ways to achieve much more flexible and efficient use of network bandwidth and server capacity than the approach illustrated in Figure 1." p. 5	The Registry and Storage Manager maintain the authoritative list of summary objects that exist in the Broker. For each object, the Registry records a time to live value and unique object identifier (OID) consisting of the object's URL plus information about the Gatherer that generated the summary object. p. 12	Dispatching is determining which Broker (page server) to use to handle the query. Brokers have different content. "For example, one Broker might index scientific papers for microbiologists, while another Broker indexes PC software archives." p. 6. See '554 1b5 for use of dynamic information.	The request must be received to be processed	Harvest uses HTTP (pp. 4, 5. 9. 11, 17-20, 25). HTTP uses TCP, hence TCP releasing	Concurrent processing is inherent because distinct machines are used	The Query Manager exports objects to the network It accepts a query, translates it into an intermediate representation, and then passes it to the search engine. The search engine responds with a list of OIDs and some search engine specific data for each For example, one of our search engines, Glimpse returns the matching SOIF attribute for each summary object in the result list." p. 13.	Brokers retrieve information from one or more Gatherers or other Brokers, and incrementally update their indexes p. 5

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
29	00W/063	Clausnitzer et al, "A WWW interface to the OMNIS/Myriad Literature Retrieval Engine," in Computer Networks and ISDN Systems, v. 27, n. 6	April 1995		'554	OMNIS generates dynamic pages from databases: "Each OMNIS-HTTP server is in permanent connection with its OMNIS database and translates incomming [sic] requests into database queries. The resulting data is then trasformed [sic] back into HTML and returned to the WWW client." Sec. 3.2. MAC000895	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format."‥ Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1
30	010	Lagoze, et al., "Dienst: Implementation Reference Manual"	May 5, 1995		'554	Dienst is a computer-implemented method for producing hypertext pages (dynamic web pages) in response to queries (requests). "The results of a search, that may have been dispatched to several servers, are presented to the user as a combined list of hyperlinks sorted by publisher and docid " p. 12	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	The Dienst Server (page server) is shown as receiving the request in Fig. 2	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." p. 11

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
29	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	Concurrency is inherent in the architecture of Fig. 5 because load-balancing is being done in passing the requests off to OMNIS servers. When the request is passed off to the OMNIS Server, the HTTP server is free to process other requests	See '554 1a	See '554 1a	The identification comes from a "global configuration file." "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3.	See '554 1a
30	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server." p. 12. The selection of page server is thus informed by this dynamic information	Concurrency is inherent in the architecture of Fig. 2. When the Web server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst allows "foreign servers" (pp. 35, 41), which run on separate machines from the Web server	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated "hit list" to the user."	"A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6	The data source is identified in the HTTP request: "The stub is a small, simple executable that strips the Dienst request from the HTTP request, packages a few important environment variables that contain information about the request (e.g., the remote host identity, the MIME types the client is willing to accept, etc.), and sends them via socket I/O to the Dienst server." p. 11. Dienst must also identify data sources to invoke "foreign servers."	See '554 1d1

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
29	Each OMNIS (page) server maintains a connection to a data source: "Each OMNIS server has a permanent connection to a single database within a permanent database session." Sec. 3.2. MAC000896			OMNIS provides a Mask Definition Language for creating such templates: "a special Mask Definition Language was developed. The masks consist of the fixed HTML fragments which are filled by the gateway application and returned to the client as request results." Sec. 3.3. MAC000896	The Gateway application contacts an OMNIS Server to interrogate a database. The retrieved data is returned to the Gateway application, which inserts it into HTML fragments in the template. See '554 7.	OMNIS is web-based, hence networked. It handles dynamic page generation requests. See '551 1a	See '554 1a	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896
30						Dienst is networked -- it uses the Internet. "The basic Dienst server interoperates with a distributed network of other Dienst servers " p.41	"A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
29	The first computer system is the "WWW Client" of Fig. 5	The second computer system includes the "HTTP Server" and "Gateway Application" of Fig. 5	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1
30	The first computer system includes the WWW Client of Fig. 2	The second computer system includes the WWW Server, CGI-Stub and Dienst Server of Fig. 2.	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." p. 11	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server." p. 12. The selection of page server is thus informed by this dynamic information	The Dienst Server (page server) is shown as receiving the request in Fig. 2

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
29	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	See '554 1a	See '554 1a	OMNIS is web-based, hence networked. It handles dynamic page generation requests. See '551 1a	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	Because the system is a programmed system, it must have a medium on which instructions are stored	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1
30	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated “hit list” to the user.”	“A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6	Dienst is networked -- it uses the Internet. "The basic Dienst server interoperates with a distributed network of other Dienst servers " p.41	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request.” p. 11	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server.” p. 12. The selection of page server is thus informed by this dynamic information	Because the system is a programmed system, it must have a medium on which instructions are stored	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	The Dienst Server (page server) is shown as receiving the request in Fig. 2

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
29	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	Concurrency is inherent in the architecture of Fig. 5 because load-balancing is being done in passing the requests off to OMNIS servers. When the request is passed off to the OMNIS Server, the HTTP server is free to process other requests	See '554 1a		OMNIS generates dynamic pages from databases: "Each OMNIS-HTTP server is in permanent connection with its OMNIS database and translates incomming [sic] requests into database queries. The resulting data is then trasformed [sic] back into HTML and returned to the WWW client." Sec. 3.2. MAC000895	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1
30	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." p. 11	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server." p. 12. The selection of page server is thus informed by this dynamic information	Concurrency is inherent in the architecture of Fig. 2. When the Web server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst allows "foreign servers" (pp. 35, 41), which run on separate machines from the Web server	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated “hit list” to the user.”		Dienst is a computer-implemented method for producing hypertext pages (dynamic web pages) in response to queries (requests). "The results of a search, that may have been dispatched to several servers, are presented to the user as a combined list of hyperlinks sorted by publisher and docid " p. 12	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	The Dienst Server (page server) is shown as receiving the request in Fig. 2

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
29	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	Concurrency is inherent in the architecture of Fig. 5 because load-balancing is being done in passing the requests off to OMNIS servers. When the request is passed off to the OMNIS Server, the HTTP server is free to process other requests	See '554 1a	See '554 1a	OMNIS generates dynamic pages from databases: "Each OMNIS-HTTP server is in permanent connection with its OMNIS database and translates incomming [sic] requests into database queries. The resulting data is then trasformed [sic] back into HTML and returned to the WWW client." Sec. 3.2. MAC000895	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"
30	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." p. 11	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	Concurrency is inherent in the architecture of Fig. 2. When the Web server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst allows "foreign servers" (pp. 35, 41), which run on separate machines from the Web server	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated “hit list” to the user.”	“A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6	Dienst is a computer-implemented method for producing hypertext pages (dynamic web pages) in response to queries (requests). "The results of a search, that may have been dispatched to several servers, are presented to the user as a combined list of hyperlinks sorted by publisher and docid " p. 12	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
29	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	The identification comes from a "global configuration file." "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3.	See '554 1a	Each OMNIS (page) server maintains a connection to a data source: "Each OMNIS server has a permanent connection to a single database within a permanent database session." Sec. 3.2. MAC000896			The mask files written in Mask Definition Language are the tag-based text templates. See '554 7	See '554 8	"The masks consist of the fixed HTML fragments which are filled by the gateway application and returned to the client as request results. These fragments also contain hidden queries for the OMNIS server and control commands for the CGI program." Sec. 3.3. "The query results are sent back to the CGI program which transforms it into a WWW client readable format." Sec. 3.2. The CGI program therefore uses the mask template to drive a format of the retrieved data
30	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server." p. 12. The selection of page server is thus informed by this dynamic information	The data source is identified in the HTTP request: "The stub is a small, simple executable that strips the Dienst request from the HTTP request, packages a few important environment variables that contain information about the request (e.g., the remote host identity, the MIME types the client is willing to accept, etc.), and sends them via socket I/O to the Dienst server." p. 11. Dienst must also identify data sources to invoke "foreign servers."	See '554 1d1						

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
29	"The masks consist of the fixed HTML fragments which are filled by the gateway application" Sec. 3.3				The method is computer-implemented. See. e.g., Abstract. MAC000888	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	The request is not processed by the Web server, but is intercepted and handed off to a page server (OMNIS Server): "The CGI program passes the WWW client's requests to one of the OMNIS servers which translate them into OMNIS queries for the database." Sec. 3.2. See also '554 1b1	The Web server is the HTTP server of Fig. 5. The page server is the OMNIS server of Fig. 5 combined with that portion of the Gateway program which generates the dynamic page. The routing is shown in Fig. 5. "Each OMNIS server has a permanent connection to a single database within a permanent database session. The CGI program passes the WWW client's request to one of the OMNIS servers which translate them into OMNIS queries for the database. The query results are sent back to the CGI program which transforms it into a WWW client readable format.".. Sec. 3.2, MAC000895; see also Fig. 5, MAC000896
30			The reference teaches managed objects: "It is an object of the present invention to process requests from Web clients forwarded by Web servers to the present invention, and to interact with other communication systems and networks, and their associated managed objects, to obtain information from the managed systems and communicate that information back to the Web client in response to the request." 4:2-8. This requires an object handling extension		The method is computer-implemented	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2	The Dienst Server (page server) is shown as receiving the request in Fig. 2	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	Interception occurs when the Web server recognizes that a request must be processed by a Dienst server: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." p. 11	The Web server is the "WWW Server" of Fig. 2 on p. 10. The page server is the Dienst Server of Fig. 2. The routing is shown in Fig. 2

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
29	Concurrency is inherent in the architecture of Fig. 5 because load-balancing is being done in passing the requests off to OMNIS servers. When the request is passed off to the OMNIS Server, the HTTP server is free to process other requests	See '554 1a	See '554 1a	The method is computer-implemented. See. e.g., Abstract. MAC000888	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	The identification comes from a "global configuration file." "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3.	See '554 1a	Each OMNIS (page) server maintains a connection to a data source: "Each OMNIS server has a permanent connection to a single database within a permanent database session." Sec. 3.2. MAC000896
30	Concurrency is inherent in the architecture of Fig. 2. When the Web server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst allows "foreign servers" (pp. 35, 41), which run on separate machines from the Web server	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated “hit list” to the user.”	“A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6	The method is computer-implemented	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server.” p. 12. The selection of page server is thus informed by this dynamic information	The data source is identified in the HTTP request: "The stub is a small, simple executable that strips the Dienst request from the HTTP request, packages a few important environment variables that contain information about the request (e.g., the remote host identity, the MIME types the client is willing to accept, etc.), and sends them via socket I/O to the Dienst server.” p. 11. Dienst must also identify data sources to invoke "foreign servers."	See '554 1d1	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
29			The mask files written in Mask Definition Language are the tag-based text templates. See '554 7	See '554 8	"The masks consist of the fixed HTML fragments which are filled by the gateway application and returned to the client as request results. These fragments also contain hidden queries for the OMNIS server and control commands for the CGI program." Sec. 3.3. "The query results are sent back to the CGI program which transforms it into a WWW client readable format." Sec. 3.2. The CGI program therefore uses the mask template to drive a format of the retrieved data	"The masks consist of the fixed HTML fragments which are filled by the gateway application" Sec. 3.3				The method is computer-implemented. See. e.g., Abstract. MAC000888
30								The reference teaches managed objects: "It is an object of the present invention to process requests from Web clients forwarded by Web servers to the present invention, and to interact with other communication systems and networks, and their associated managed objects, to obtain information from the managed systems and communicate that information back to the Web client in response to the request." 4:2-8. This requires an object handling extension		The method is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
29	The dispatcher is the "Gateway Program" of Fig. 5, which shows the required routing as "Gateway Program" joined to "HTTP Server"	The dynamic information is contained in configuration files, which may be updated at any time. "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses and defines how resulting HTML pages are to be generated." Section 3.3, MAC000896. This is configuration file dispatching	Each OMNIS Server accesses a different database, so the system must determine which database is needed to fulfill the request and send the query to correct server. This process is informed by a configuration file: "The CGI program analyzes the client's request and reads a global configuration file. This file contains OMNIS server addresses [sic] and defines how resulting HTML pages are to be generated." Sec. 3.3. The reference also teaches load balancing: "In the future an automatic load balancing of OMNIS servers for one database would be easy to implement." Sec. 3.2	Receiving is shown in Fig. 5 as a double arrow between "Gateway Program" and "OMNIS Server." See also '554 1b1	Fig. 5 explicitly shows TCP communications, so TCP releasing occurs per EpicRealm's contentions	Concurrency is inherent in the architecture of Fig. 5 because load-balancing is being done in passing the requests off to OMNIS servers. When the request is passed off to the OMNIS Server, the HTTP server is free to process other requests	See '554 1a	See '554 1a
30	The dispatcher comprised the CGI stub and that component of a Dienst server which recognizes the need to request content from a different Dienst server. "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers." p. 11. The CGI stub and Dienst server send the request to the appropriate destination	See '554 1b6	A "meta-server" maintains dynamic information about page servers: "Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server." p. 12. The selection of page server is thus informed by this dynamic information	The Dienst Server (page server) is shown as receiving the request in Fig. 2	Requests are sent "via socket I/O to the Dienst server." p. 11. Therefore the Web server is released via SBL releasing per EpicRealm's contention. "Since Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the WorldWide Web) as a transport layer for requests." Therefore the Web server is released via TCP releasing per EpicRealm's contention	Concurrency is inherent in the architecture of Fig. 2. When the Web server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst allows "foreign servers" (pp. 35, 41), which run on separate machines from the Web server	One example is on p. 12: "Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated “hit list” to the user.”	“A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." p. 6

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
31	161	Scholl et al. U.S. Patent 5,742,762	May 19, 1995		'554	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server" Abstract	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	The request must be received to be processed.	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." 2:42-44
32	607	Introduction to Cold Fusion	May 23, 1995		'554	"You will be able to insert data submitted from HTML forms to tables in your database as well as allow users to submit queries with flexible criteria and display the results in attractively formatted table or any other output format which you wish." ORCL01038523. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	The request must be received to be processed	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
31	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which "appropriate network proxy agent" should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	Concurrency is inherent in the architecture of Fig. 3	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")	Network management system/managed network 6 identifies one or more data sources from which to retrieve the data using Management Information Base (MIB) 14 or the network management proxy agents 15. 6:57 7:30	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML" Abstract
32	Single-server dispatching	Single-server dispatching	Concurrency is shown at ORCL01038534: "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." The "launching an instance" means the Web server may continue handling requests	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524	The data source is identified by the DATASOURCE DBML tag. ORCL01038544	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
31			"Then network management information transmissions are received 28 in response to each NMR from a managed network (and may be stored in the local database for future retrieval)." 7:64-67. This is page caching because the data in the information transmissions is displayable on a browser and thus constitutes a page under EpicRealm's Partial Page Server contention			"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server, appearing to the client as information in a Web file." Abstract	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3. Under an alternative theory based on EpicRealm's contentions, the "page server having a processing means" is the network management system 9 of each managed network 6. Even though the network management system 9 does not produce finished pages, its output is displayable by a browser and hence, according to EpicRealm, constitutes a web page. Therefore, network management system 9 can be considered a page server.
32		"Under Windows NT, ODBC data source information is user-account specific. Meaning that if you define a data source while looged in as one user it will not be available to other users." ORCL8561		"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523 "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	Cold Fusion is a networked system for managing a dynamic Web page generation request. See '554 1a	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
31	The “first computer system including means for generating said request” of element 9d is the web client 1 of Fig. 3	The second computer system is the web server 3	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the “output formatter” 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the “output formatter” 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, “Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways.” 2:42-44	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which “appropriate network proxy agent” should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	The request must be received to be processed.
32	The first computer system includes the user's web browser ("Remote Web Client" in the figure on ORCL01038534)	The second computer system includes the "Web server" in the figure on ORCL01038534	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	Single-server dispatching	Single-server dispatching	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
31	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server, appearing to the client as information in a Web file." Abstract	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." 2:42-44	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which "appropriate network proxy agent" should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	Because the system is a programmed system, it must have a medium on which instructions are stored	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	The request must be received to be processed.
32	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524	Cold Fusion is a networked system for managing a dynamic Web page generation request. See '554 1a	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	Single-server dispatching	Because the system is a programmed system, it must have a medium on which instructions are stored.	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
31	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." 2:42-44	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which "appropriate network proxy agent" should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	Concurrency is inherent in the architecture of Fig. 3	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)		"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server" Abstract	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	The request must be received to be processed.
32	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	Single-server dispatching	Single-server dispatching	Concurrency is shown at ORCL01038534: "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." The "launching an instance" means the Web server may continue handling requests	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	'335	"You will be able to insert data submitted from HTML forms to tables in your database as well as allow users to submit queries with flexible criteria and display the results in attractively formatted table or any other output format which you wish." ORCL01038523. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
31	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." 2:42-44	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	Concurrency is inherent in the architecture of Fig. 3	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML, and transmits this information to the Web client via the Web server" Abstract	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3
32	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	Concurrency is shown at ORCL01038534: "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." The "launching an instance" means the Web server may continue handling requests	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524	"You will be able to insert data submitted from HTML forms to tables in your database as well as allow users to submit queries with flexible criteria and display the results in attractively formatted table or any other output format which you wish." ORCL01038523. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	Single-server dispatching

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
31	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which “appropriate network proxy agent” should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	Network management system/managed network 6 identifies one or more data sources from which to retrieve the data using Management Information Base (MIB) 14 or the network management proxy agents 15. 6:57-7:30	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML" Abstract			"Then network management information transmissions are received 28 in response to each NMR from a managed network (and may be stored in the local database for future retrieval)." 7:64-67. This is page caching because the data in the information transmissions is displayable on a browser and thus constitutes a page under EpicRealm's Partial Page Server contention			
32	Single-server dispatching	The data source is identified by the DATASOURCE DBML tag. ORCL01038544	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523		"Under Windows NT, ODBC data source information is user-account specific. Meaning that if you define a data source while looged in as one user it will not be available to other users." ORCL8561		"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"You will be able to insert data submitted from HTML forms to tables in your database as well as allow users to submit queries with flexible criteria and display the results in attractively formatted table or any other output format which you wish." "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
31			The reference teaches managed objects: "It is an object of the present invention to process requests from Web clients forwarded by Web servers to the present invention, and to interact with other communication systems and networks, and their associated managed objects, to obtain information from the managed systems and communicate that information back to the Web client in response to the request." 4:2-8. This requires an object handling extension		The method is computer-implemented	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3	The request must be received to be processed.	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	Intercepting occurs when software in the web server 3 determines that the request is not to be handled by the web server but is instead sent to the network management gateway 5. 6:9-15. The reference teaches, "Also, Web servers can forward client requests for data, that neither the Web client nor the Web server can access directly, to applications called gateways." 2:42-44	The Web server is web server 3 of Fig. 3. The page server includes the network management system 9 of each managed network 6 and the "output formatter" 17 of gateway 5. Figs. 3, 4. The routing is shown in Fig. 3
32	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524. UPDATE is an SQL primitive. "This Chapter explains how to use Cold Fusion and the DBML Engine to insert and update data." ORCL01038536			Cold Fusion is computer-implemented	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534	The request must be received to be processed	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	Intercepting is determining that the DBML Engine is to be invoked rather than the Web server. "Each HTML form which create has an ACTION attribute which instructs the web server which CGI compliant script application should be executed to process the form submission. This application is then executed and passed the contents of the form ... In order to fulfill the request, the application must then either route the client to another URL or dynamically generate an HTML page to be returned to the client." ORCL01308528	The Web server of the Patent is the Web server of the reference. The page server is the "DBML Engine." "All client requests are processed using the DBML Engine, a CGI script provided with Cold Fusion (DBML.EXE). This script handles interacting with the Web server and database as well as dynamically generating HTML to be sent back to the client." ORCL01038528. The routing is shown in the figure on ORCL01038534. "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." ORCL01038534

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
31	Concurrency is inherent in the architecture of Fig. 3	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")	The method is computer-implemented	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which "appropriate network proxy agent" should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	Network management system/managed network 6 identifies one or more data sources from which to retrieve the data using Management Information Base (MIB) 14 or the network management proxy agents 15. 6:57-7:30	"The present invention receives requests from a Web client forwarded by a Web server and interacts with the managed networks and their associated objects to obtain information. The present invention then converts this information in real time to hypermedia document format in HTTP and HTML" Abstract	
32	Concurrency is shown at ORCL01038534: "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." The "launching an instance" means the Web server may continue handling requests	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524	Cold Fusion is computer-implemented	Single-server dispatching	Single-server dispatching	The data source is identified by the DATASOURCE DBML tag. ORCL01038544	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
31		"Then network management information transmissions are received 28 in response to each NMR from a managed network (and may be stored in the local database for future retrieval)." 7:64-67. This is page caching because the data in the information transmissions is displayable on a browser and thus constitutes a page under EpicRealm's Partial Page Server contention						The reference teaches managed objects: "It is an object of the present invention to process requests from Web clients forwarded by Web servers to the present invention, and to interact with other communication systems and networks, and their associated managed objects, to obtain information from the managed systems and communicate that information back to the Web client in response to the request." 4:2-8. This requires an object handling extension		The method is computer-implemented
32	"Under Windows NT, ODBC data source information is user-account specific. Meaning that if you define a data source while looged in as one user it will not be available to other users." ORCL8561		"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"You will be able to insert data submitted from HTML forms to tables in your database as well as allow users to submit queries with flexible criteria and display the results in attractively formatted table or any other output format which you wish." "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "To develop a Cold Fusion application, you create pages which contain both HTML and DBML tags (these pages arfe referred to as templates) ... the Cold Fusion DBML Engine (DBML.EXE) processes this template file, executing any queries contained in it and formatting the query results as specified." ORCL01038529	"The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524. UPDATE is an SQL primitive. "This Chapter explains how to use Cold Fusion and the DBML Engine to insert and update data." ORCL01038536			Cold Fusion is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
31	The dispatcher is the network management gateway. Routing is shown as an arrow from box 18 to box 11 in Fig. 3	The network management proxy agents 15 keep information about the data sources available on the managed networks 6. 7:5-35; 7:49-8:10. The Management Information Database (MIB) 14 of Fig. 4 maintains "a database of information regarding the managed networks 6 and information regarding the network management gateway 5." 7:2-5.	Dispatching occurs in block 25 of Fig. 6 ("Forward Request to Appropriate Network management Proxy Agent") when the network management gateway determines which "appropriate network proxy agent" should receive the request (i.e. makes an informed selection) and forwards the request to that agent. The selection is informed because an "appropriate" agent is chosen	The request must be received to be processed.	"the Web server 3 forwards the request to the present invention 5, communicating through the Internet or otherwise." 6:13-15. The mode of Internet communication taught in the reference is HTTP (Abstract), which performs TCP releasing according to EpicRealm's contentions. This is in accord with the implication of the Patents that the dispatcher is part of the Web server.	Concurrency is inherent in the architecture of Fig. 3	The response to the request is converted "in real time to hypermedia document format in HTTP and HTML." (Abstract)	The Web page includes data dynamically retrieved from one or more data sources of the managed network system(s) 9. Fig. 3. The data sources are the managed network elements 8. Also Fig. 6 ("receive information from managed network")
32	Single-server dispatching	See "Setting Up ODBC Data Sources" at ORCL01038531. The collection of sources can be changed dynamically	Single-server dispatching	The request must be received to be processed	Per EpicRealm's contentions, the web server is freed when it launches an instance of the DBML Engine (page server). ORCL01038534	Concurrency is shown at ORCL01038534: "Web server launches an instance of DBML.EXE (the DBML Engine) passing it the data submitted by the client." The "launching an instance" means the Web server may continue handling requests	"Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523	The "data sources" are, e.g. the "ODBC Data Source" in the figure on ORCL01038534. "Rather than writing code in a traditional programming language you instead embed DBML (Database Markup Language) tags inside standard HTML documents. The Cold Fusion DBML Engine then interprets and executes these commands,, converts the results to HTML, and sends the results back to the client." ORCL01038523. "The ability to send any valid SQL statement to the database and embed parameters from Form submissions, URL query strings, and HTTP environment headers in the SQL statement." ORCL01038524

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
33	162	Lagarde et al. 5,745,754	Jun. 7, 1995		'554	"[O]ur control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet. Our control program agents using the CGI interface can create HTML commands dynamically. In this way a program can present information on a Web browser for the Web client." 8:2:5	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The request must be received to be processed.	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73
34	180	Rogers et al. 5,701,451	Jun. 7, 1995		'554	The reference teaches "dynamically creat[ing] new HTML statements for a report of results according to a request of the web browser" to a Web server. Claims 1, 16	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Control Program Agent and DIS (page server) receive the request. See Fig. 1	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13-19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
33	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the [	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking." Abstract
34	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely…." 4:55-59

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
33	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16	Logging in to a data source is expressly taught in the reference. Step 115 of Fig. 8 shows a “DIS logon.” It is described at 14:9-13: “At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS "desktop", our DIS file server 14.”				"A World Wide Web browser makes requests to web servers on a network which receive and fulfill requests as an agent of the browser client, organizing distributed sub-agents as distributed integration solution (DIS) servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking." Abstract	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.
34	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16	"It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any "hidden" default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the "hidden" password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet." 10:48-56. Passwords are used for logging in.				The title of the reference is "Method for fulfilling requests of a web browser," which teaches a networked system	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
33	The first computer system is the web client 71 (web browser) of Fig. 7.	The second computer system includes Web server 72, control program agent 73 and ODAS 74 of Fig. 7	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to	The request must be received to be processed.
34	The first computer system is the web client 71 (web browser) of Fig. 7.	The second computer system includes Web server 72, control program agent 73 and ODAS 74 of Fig. 7	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	The Control Program Agent and DIS (page server) receive the request. See Fig. 1

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
33	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	Generation of a dynamic page is described at 4:52-55	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	"A World Wide Web browser makes requests to web servers on a network which receive and fulfill requests as an agent of the browser client, organizing distributed sub-agents as distributed integration solution (DIS) servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations,	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the DIS Cap	Because the system is a programmed system, it must have a medium on which instructions are stored	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The request must be received to be processed.
34	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13-19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	The title of the reference is "Method for fulfilling requests of a web browser," which teaches a networked system	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	Because the system is a programmed system, it must have a medium on which instructions are stored	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Control Program Agent and DIS (page server) receive the request. See Fig. 1

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
33	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the DIS Cap	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55	'335	"[O]ur control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet. Our control program agents using the CGI interface can create HTML commands dynamically. In this way a program can present information on a Web browser for the Web client." 8:2:5	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The request must be received to be processed.
34	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13-19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	'335	The reference teaches "dynamically creat[ing] new HTML statements for a report of results according to a request of the web browser" to a Web server. Claims 1, 16	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Control Program Agent and DIS (page server) receive the request. See Fig. 1

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
33	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	“[O]ur control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet. Our control program agents using the CGI interface can create HTML commands dynamically. In this way a program can present information on a Web browser for the Web client.” 8:2:5	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher
34	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13-19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS “objects,” described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. “The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client….” 5:45-51	The reference teaches “dynamically creat[ing] new HTML statements for a report of results according to a request of the web browser” to a Web server. Claims 1, 16	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
33	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to qu	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking." Abstract	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16	Logging in to a data source is expressly taught in the reference. Step 115 of Fig. 8 shows a "DIS logon." It is described at 14:9-13: "At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS "desktop", our DIS file server 14."		The tag-based (HTML) text templates are provided by the CGI application: "In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server. After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:57-8:2	See '335 8	After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:66-8:2
34	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65 16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely…." 4:55-59	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16	"It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any "hidden" default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the "hidden" password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet." 10:48-56. Passwords are used for logging in.		The tag-based (HTML) text templates are provided by the CGI application: "In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server. After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:33-45	See '335 8	"After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet."

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
33	The tag-based templates are HTML templates	"This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retrie[ve]d from a database to be performed." 13:30-33. "Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved." 16:2-4	The reference teaches "an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document" Abstract. The access to the selected capsule objects is through an object handling extension		The method is computer-implemented	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The request must be received to be processed.	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	The handling of the request is intercepted from the web server and not processed by it, but instead is sent to the control program agent 73	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.
34	The tag-based templates are HTML templates	"This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retrie[ve]d from a database to be performed." 13:30-33. "Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved." 16:2-4	The reference teaches "an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document" Abstract. The access to the selected capsule objects is through an object handling extension		The method is computer-implemented	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.	The Control Program Agent and DIS (page server) receive the request. See Fig. 1	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13 19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Intercepting occurs because the request is deflected from the web server and not processed by it, but instead is sent to the control program agent 73.	The Web server is box 72 in Fig. 7. The page server is a combination of the DIS 75 (DIS "objects," described at 17:6) and the control program agent 73. Routing from the Web server to the page server occurs because the request is routed (sent) from the web server 72 to the DIS 75.

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
33	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51	The method is computer-implemented	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODA	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking." Abstract	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16
34	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51	The method is computer-implemented	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	The DIS identifies the data sources (databases) from which to retrieve data. 13:1-10.	"Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely…." 4:55-59	The DIS maintains a set of connections to multiple databases, which constitutes a connection cache. 12:9-16

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
33	Logging in to a data source is expressly taught in the reference. Step 115 of Fig. 8 shows a “DIS logon.” It is described at 14:9-13: “At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS "desktop", our DIS file server 14.”		The tag-based (HTML) text templates are provided by the CGI application: "In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server. After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:57-8:2	See '335 8	After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:66-8:2	The tag-based templates are HTML templates	“This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retrie[ve]d from a database to be performed.” 13:30-33. “Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved.” 16:2-4	The reference teaches "an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document" Abstract. The access to the selected capsule objects is through an object handling extension		The method is computer-implemented
34	"It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any "hidden" default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the "hidden" password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet." 10:48-56. Passwords are used for logging in.		The tag-based (HTML) text templates are provided by the CGI application: "In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server. After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:33-45	See '335 8	"After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet."	The tag-based templates are HTML templates	“This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retrie[ve]d from a database to be performed.” 13:30-33. “Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved.” 16:2-4	The reference teaches "an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document" Abstract. The access to the selected capsule objects is through an object handling extension		The method is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
33	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	"FIG. 7 shows the coupling within the Web server from ODAS 74 to a distributed DIS LAN 75 with heterogeneous connections to multiple databases DB2, Redbrick, Sybase and Oracle." 12:13-17. ODAS must maintain dynamic information about the databases to which it is connected so it can select the correct database and communicate with it to process a request	An informed selection of DIS object is made based on the nature of the content to be created (15:65-16-62) and dynamic information maintained about page servers: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. ... the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 13:7-19. "the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the DIS Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the DIS Cap	The request must be received to be processed.	TCP/IP is shown as the mechanism for sending to the DIS in Fig. 11 and is described at 19:35-41. Therefore, the releasing is TCP releasing according to EpicRealm's contentions.	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51
34	The request is routed (sent) to the ODAS 74, which dispatches it to the appropriate DIS (page server). Hence ODAS is the dispatcher	"FIG. 7 shows the coupling within the Web server from ODAS 74 to a distributed DIS LAN 75 with heterogeneous connections to multiple databases DB2, Redbrick, Sybase and Oracle." 12:13-17. ODAS must maintain dynamic information about the databases to which it is connected so it can select the correct database and communicate with it to process a request	ODAS makes an informed selection of which DIS the request should be sent to. The particular choice of object is made based on the nature of the content to be created, e.g. text, an Excel spreadsheet or graphics 15:65-16-62. Dynamic information about page servers is employed: "The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired." 12:49-61. See also Lagarde ref. 162	The Control Program Agent and DIS (page server) receive the request. See Fig. 1	TCP/IP is shown as the mechanism for sending the request to the DIS (page server) in Fig. 11 and is described at 19:13-19. Therefore, the releasing is TCP releasing according to EpicRealm's contentions	Web server 72 processes other requests while the page generation request has been passed off to the DIS. The concurrency is shown in Figs. 1 and 11	Generation of a dynamic page is described at 4:52-55. Also, "After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet." 7:42-45	The page includes data from one or more of the databases identified in Fig. 7, i.e. Sybase, Oracle, DB2 and RedBrick. "The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client…." 5:45-51

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
35	194	Popp et al. U.S. Patent 6,651,108, "Method and Apparatus for Generating Object-Oriented World Wide Web Pages"	Aug. 14, 1995			"The object-oriented approach used by the present invention provides the ability to dynamically generate a page, or HTML document. Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:63-6:1	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24
36	195	Popp et al. 6,249,291	Sept. 22, 1995		'554	Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). Abstract	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	The request must be received to be processed	TCP releasing	Intercepting is what sends the request to the CGI program 322 instead of the web server

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
35	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	Concurrency is inherent in the architecture. The patent incorporates reference 195 by reference, so see Popp '291 below for this element	See '554 1a, Fig. 3 and 8:45-9:9.	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1	"For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input. The form names can include the name of the form that was submitted to initiate the client request as well as a return form. The application name identifies the application that services the client request." 6:49-7:5	See '554 1d2 and 7:63-8:6
36	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:24-35	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	The web server and page server are different machines, so concurrent processing occurs inherently. "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:25-35.	The application 214 returns "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." 7:44-58.	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46	CGI server 906 identifies the Web Object server 912 based on the information contained in the request, for example. 26:3-4	For example, the present invention can be used to access a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic. 7:45-49

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
35				"At step 302, a Web page developer creates the HTML templates to provide a Web page definition." 8:44-45. "When the Web page definition is requested by a client browser, the associated HTML template(s), script(s), and declarations are retrieved from storage. At step 306, a parser is invoked to parse the statements contained in the HTML template(s)." 8:54-58. "The present invention provides an extension to HTML that is used on the HTTP server side." 10:47-48. "The present invention uses an extension to the standard HTML known as the group extension The group extension provides the ability to combine HTML elements or statements in a single block." 10:57-60	"The STRING HTML object is inserted into the group identified by the entry. When the HTML document is rendered, the STRING HTML object generates the HTML statements necessary to insert the text string into the HTML document." 4:30-34. "An HTML object is instantiated for each of the HTML elements defined in the HTML template(s) at step 308. At step 310, the objects instantiated in step 308 are used to render the Web page definition (i.e., HTML document)." 8:62-65	The disclosed system is Web-based, hence networked. See also '554 1	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25
36	State information can be retained to allow the internal application to allow the application to maintain open sessions with multiple users capable of processing multiple transactions in any order. 554 8:28-31. "For example, a connection to a database can be used at the global level by multiple sessions and applications." 13:38-40.	A digital signature can be used as a key to perform verification. For example, the digital signature can be used to verify that a request is a valid request generated by a valid client. The digital signature can further be used to verify that the client is authorized to access the data of the specified session. 27:12-17		The present invention provides an extension to HTML that is used on the HTTP server side. The HTML extension is filtered out before a Web page is sent to a client browser. It is used to interpret an HTML template and to render an HTML document before is transmitted to the client browser 15:45-50	A stringcontrol object instance instantiates an NSWHString static HTML element and inserts it in the template. 18:57-59	Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). Abstract. Networked: "The present invention provides the ability to access an application using any interface. For example, a client can access the same application via the Internet using a system running Windows, MAC/OS, Sun.OS, NextStep, etc. Referring to FIG. 2, client 202 is resident on a corporate network 200. Corporate network 200 is a local area network comprised of personal computers such as client 202, for example." 6:32-40	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
35	The first computer system includes the client browser 202 of Fig. 2. See 6:29-34	The second computer system includes the HTTP Server 206, the CGIMessenger 210 and the Application 214 of Fig. 2	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25
36	First computer system is the client. "Requests submitted by clients 302, 304, 306, and 308 to HTTP server 314 are transmitted to HTTP server 314 via WWW 310." 8:17-19.	The second computer system is box 208 in Fig. 2 and everything to its right	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	Intercepting is what sends the request to the CGI program 322 instead of the web server	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:24-35	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
35	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	See '554 1a, Fig. 3 and 8:45-9:9.	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1	The disclosed system is Web-based, hence networked. See also '554 1	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	The invention is a computer system, which requires instructions to be stored on a machine-readable medium. See also 5:12-49	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25
36	TCP releasing	The application 214 returns “a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic.” 7:44-58.	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46	Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). Abstract. Networked: "The present invention provides the ability to access an application using any interface. For example, a client can access the same application via the Internet using a system running Windows, MAC/OS, Sun.OS, NextStep, etc. Referring to FIG. 2, client 202 is resident on a corporate network 200. Corporate network 200 is a local area network comprised of personal computers such as client 202, for example." 6:32-40	Intercepting is what sends the request to the CGI program 322 instead of the web server	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	Because the system is a programmed system, it must have a medium on which instructions are stored	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
35	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	Concurrency is inherent in the architecture. The patent incorporates reference 195 by reference, so see Popp '291 below for this element	See '554 1a, Fig. 3 and 8:45-9:9.		"The object-oriented approach used by the present invention provides the ability to dynamically generate a page, or HTML document. Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:63-6:1	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25
36	TCP releasing	Intercepting is what sends the request to the CGI program 322 instead of the web server	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. “The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server).” 7:24-35	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	The web server and page server are different machines, so concurrent processing occurs inherently. "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:25-35.	The application 214 returns “a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic.” 7:44-58.	'335	Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). Abstract	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
35	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	Concurrency is inherent in the architecture. The patent incorporates reference 195 by reference, so see Popp '291 below for this element	See '554 1a, Fig. 3 and 8:45-9:9.	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1	"The object-oriented approach used by the present invention provides the ability to dynamically generate a page, or HTML document. Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:63-6:1	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.
36	TCP releasing	Intercepting is what sends the request to the CGI program 322 instead of the web server	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	The web server and page server are different machines, so concurrent processing occurs inherently. "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:25-35.	The application 214 returns "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." 7:44-58.	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46	Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). Abstract	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:24-35

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
35	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	"For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input. The form names can include the name of the form that was submitted to initiate the client request as well as a return form. The application name identifies the application that services the client request." 6:49-7:5	See '554 1d2 and 7:63-8:6				HTML templates are tag-based. See '554 7 and 7:33-62	See '554 8	"A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." 2:26-28. Thus the tag-based (HTML) text templates drive a format of the data
36	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	CGI server 906 identifies the Web Object server 912 based on the information contained in the request, for example. 26:3-4	For example, the present invention can be used to access a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic. 7:45-49	State information can be retained to allow the internal application to allow the application to maintain open sessions with multiple users capable of processing multiple transactions in any order. 554 8:28-31. "For example, a connection to a database can be used at the global level by multiple sessions and applications." 13:38-40.	A digital signature can be used as a key to perform verification. For example, the digital signature can be used to verify that a request is a valid request generated by a valid client. The digital signature can further be used to verify that the client is authorized to access the data of the specified session. 27:12-17		Any reference with HTML templates (e.g. this one) has tag-based text templates	Each template contains a type, a Uniform Resource Locator (URL), and a list of byte count delimited attribute value pairs. p. 6. Also Fig. 3. A stringcontrol object instance instantiates an NSWHString static HTML element and inserts it in the template. 18:57-59	The tag-based text templates are HTML templates, which drive a format of the data: "A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." 2:26-28

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
35	See '554 7. Also, "The HTML elements that define a Web page can be included in a single ASCII file. A file that contains HTML elements is referred to as an HTML template. An HTML template can include a complete definition of the Web page. However, an HTML template may define a subset of the Web page. Two or more HTML templates that define portions of the Web page can be combined to provide a complete definition." 7:29-37	"A declaration entry modifies its associated group by adding element to the group or modifying the elements that already exist in the group. For example, an HTML object, or element, declared in the entry inserts itself into the group that bears the same name as the declaration entry. Property values that are declared in the declaration entry are used to modify the HTML object's properties." 4:9-15. See also claim 6: "The method of claim 5 wherein said information is used to update an external data source."	The invention is object-oriented: "For example, a root object (e.g., a PAGE object) is instantiated at runtime. During processing, additional HTML objects can be instantiated at inserted into the object tree. ... Additional HTML objects can be added to the object tree in a similar manner until all of the objects have been assembled." 3:41-50		The invention is computer-implemented	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	Intercepting the handling of the request occurs because the request is not handled by the Web server but is routed to the page server. See 7:14-24	The Web server is the HTTP Server 206 of Fig. 2. The page server is the application 214 of Fig. 2. The routing is shown in Fig. 2. See also 6:25-7:25
36	Any reference with HTML extension templates (e.g. this one) has HTML templates	Claim 22: "computer readable program code configured to cause a computer to update said external data source with said object data"	The reference teaches generating dynamic pages through objects: "All or some portion of the Web page is dynamically generated. Classes of object are associated with elements that defined a Web page (e.g., HTML elements). By objectifying Web pages it is possible to gain the advantages of object-oriented programming with Web pages." 4:5-9. This is accomplished through object handling extensions		The present invention can be implemented on a general purpose computer such as illustrated in FIG. 1. 5:56-57	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21	The request must be received to be processed	TCP releasing	Intercepting is what sends the request to the CGI program 322 instead of the web server	Page server = application server 316. "A client request is forwarded to application server 316 via HTTP server 314." 8:20-21

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
35	Concurrency is inherent in the architecture. The patent incorporates reference 195 by reference, so see Popp '291 below for this element	See '554 1a, Fig. 3 and 8:45-9:9.	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1	The invention is computer-implemented	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	"For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input. The form names can include the name of the form that was submitted to initiate the client request as well as a return form. The application name identifies the application that services the client request." 6:49-7:5	See '554 1d2 and 7:63-8:6	
36	The web server and page server are different machines, so concurrent processing occurs inherently. "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:25-35.	The application 214 returns "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." 7:44-58.	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46	The present invention can be implemented on a general purpose computer such as illustrated in FIG. 1. 5:56-57	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:24-35	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	CGI server 906 identifies the Web Object server 912 based on the information contained in the request, for example. 26:3-4	For example, the present invention can be used to access a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic. 7:45-49	State information can be retained to allow the internal application to allow the application to maintain open sessions with multiple users capable of processing multiple transactions in any order. 554 8:28-31. "For example, a connection to a database can be used at the global level by multiple sessions and applications." 13:38-40.

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
35			HTML templates are tag-based. See '554 7 and 7:33-62	See '554 8	"A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." 2:26-28. Thus the tag-based (HTML) text templates drive a format of the data	See '554 7. Also, "The HTML elements that define a Web page can be included in a single ASCII file. A file that contains HTML elements is referred to as an HTML template. An HTML template can include a complete definition of the Web page. However, an HTML template may define a subset of the Web page. Two or more HTML templates that define portions of the Web page can be combined to provide a complete definition." 7:29-37	"A declaration entry modifies its associated group by adding element to the group or modifying the elements that already exist in the group. For example, an HTML object, or element, declared in the entry inserts itself into the group that bears the same name as the declaration entry. Property values that are declared in the declaration entry are used to modify the HTML object's properties." 4:9-15. See also claim 6: "The method of claim 5 wherein said information is used to update an external data source."	The invention is object-oriented: "For example, a root object (e.g., a PAGE object) is instantiated at runtime. During processing, additional HTML objects can be instantiated at inserted into the object tree. ... Additional HTML objects can be added to the object tree in a similar manner until all of the objects have been assembled." 3:41-50		The invention is computer-implemented
36	A digital signature can be used as a key to perform verification. For example, the digital signature can be used to verify that a request is a valid request generated by a valid client. The digital signature can further be used to verify that the client is authorized to access the data of the specified session. 27:12-17		Any reference with HTML templates (e.g. this one) has tag-based text templates	Each template contains a type, a Uniform Resource Locator (URL), and a list of byte count delimited attribute value pairs. p. 6. Also Fig. 3. A stringcontrol object instance instantiates an NSWHString static HTML element and inserts it in the template. 18:57-59	The tag-based text templates are HTML templates, which drive a format of the data: "A language is used to define the format for a page to be displayed in the WWW. The language is called Hypertext Markup Language (HTML)." 2:26-28	Any reference with HTML extension templates (e.g. this one) has HTML templates	Claim 22: "computer readable program code configured to cause a computer to update said external data source with said object data"	The reference teaches generating dynamic pages through objects: "All or some portion of the Web page is dynamically generated. Classes of object are associated with elements that defined a Web page (e.g., HTML elements). By objectifying Web pages it is possible to gain the advantages of object-oriented programming with Web pages." 4:5-9. This is accomplished through object handling extensions		The present invention can be implemented on a general purpose computer such as illustrated in FIG. 1. 5:56-57

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
35	The dispatcher is the CGIMessenger 210 of Fig. 2. The routing is shown in Fig. 2. The choice of application (page server) is informed: "To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains the information that accompanied the user request such as form and/or application names and/or user input." 6:63-7:1. "The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 204 to application 214. Application 214 executes an interaction flow to satisfy the user request." 7:4-9. The request is sent to the page server.	"A client request can result in the invocation of application 214 in the server domain 208 for example. The client request is received by the HTTP server 206. The HTTP server 206 initiates CGIMessenger 210. To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request." 6:60-65. The dynamic information comes from the HTTP server	"For requests other than simple hypertext requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application includes objects such as those created in an objective C, C++, or SmallTalk environment to manage the components of an HTML document (i.e., HTML elements). The behavior defined by the objects manipulate the HTML elements based on the information available to the objects, for example." 7:12-20. The "information available to the objects" is dynamic information, thus the selection of application (page server) is based on dynamic information maintained about page servers	The page server must receive the request in order to process it. Receiving is shown in Fig. 2. See also 6:25-7:25	The request is sent to the page server via CGI. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to EpicRealm's contention. See 6:38-7:13	Concurrency is inherent in the architecture. The patent incorporates reference 195 by reference, so see Popp '291 below for this element	See '554 1a, Fig. 3 and 8:45-9:9.	"Some or all of a page can be generated based on input received in a receive form, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." 5:65-6:1
36	CGI program 322 is the dispatcher. "CGI program 322 passes the shopper's input contained in the "Welcome" page to the application 320 running on application server 316 via corporate network 312." 9:41-44. “The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:24-35	“At step 502, the request is received by the Web Objects server. Using the information contained in the request, the Web Objects server identifies the application that will process the request. At step 504, the Web Objects server determines whether the application is executing either on the same or a different server, for example. If the application is not currently loaded, the Web Objects server initiates the application at step 508 and processing continues at step 506. If the application is executing, processing continues to step 506.” 27:45-54	See '554 1b5. " For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request. The application preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW." 7:50-58.	The request must be received to be processed	TCP releasing	The web server and page server are different machines, so concurrent processing occurs inherently. "If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server)." 7:25-35.	The application 214 returns “a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic.” 7:44-58.	The contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318). 8:44-46

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
37	501/502	"Installation and Planning Guide for Microsoft Internet Information Server Version 1.0 Beta 2" and "ODBC Web Database Add-on for Microsoft IIS Beta"	Sept. 27, 1995		'554	"Your Internet Server can contain anything from static pages of information to interactive programs that take data and return customized pages for remote users." MS-SUB EPIC000092. "[w]ith the Web Server and ODBC, you can create Web pages with information contained in a database," thus enabling generation of dynamic Web pages. (MS-SUB EPIC 000004, 000097)	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between "WWW service" and "Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request ("intercepting" it) and hands it off to the page server. (See MS-SUB EPIC 000007-0000010; 000097-000098).
38	401	Anand et al. U.S. Patent 5,692,181, "System and method for generating reports from a computer database"	Oct. 12, 1995		'554	The reference discloses a Web system for generating dynamic documents: "An Abstract Query contains placeholders for Business Concepts and Business Indicators that are filled in with the user selections in the process of instantiation, as well as conditions that are evaluated to generate different output and actual layout information. The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:36-45	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The page server must receive the request to process it. Receipt is shown in Fig. 1.	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind-Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
37	The dispatcher is that component of the Database Connector which reads the WDG file. The URL of a request will reference the Database Connector (Httpodbc.dll) as well as a "WDG file" that "contain[s] the necessary information to connect to the desired ODBC data source …" (See MS-SUB EPIC 000005; 000098) Using this information, a component of the Web server of the MS IIS invokes the Database Connector and passes the request parameters to it.	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. "Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs." (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	"Httpodbc.dll obtains the name of the WDG file [specified in the request]. . .Httpodbc.dll reads the WDG file . . . Httpodbc.dll connects to the ODBC data source, and executes the SQL statement contained within the DG file . . ." (MS-SUB EPIC 000009). IIS supports multiple simultaneous (concurrent) connections. (See MS-SUB EPIC 000071-000072).	This function is performed by httpodbc.dll. See also '554 1d2.	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.	The WDG file referenced in the URL of a request for a dynamic Web page includes information identifying the "data source" from which data is to be retrieved. See the sample file on page MS-SUB EPIC 000008	The page server (httpodbc.dll) "connects to the ODBC data source" and executes SQL statements against the data source to obtain data for the dynamic web page. MS-SUB EPIC 000009
38	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	"Thus, system 10 is implemented as a three-tier architecture." 5:48-49. Concurrency is inherent in a three-tier architecture. "Server computer 32 is preferably a multi-processor computer" 5:64-65. Concurrency is inherent in a multiprocessor architecture	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:40-45	See '554 1d1	"When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14." 5:41-43. The required identification is based on the list of Alerts and Scheduled Smart Report Requests	"DSM subsystem 16 reads schema from data warehouse 24, creates data views, and creates a mapping between the two. It also uses that mapping to translate the Dimensional Queries received from DAI subsystem 14 into SQL and package and return the results." 5:32-36. The data is retrieved via an SQL query

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
37	The diagram on page MS-SUB EPIC000098 shows that ODBC can connect simultaneously to multiple databases. (It must handle concurrent requests which require different databases.) For these simultaneous connections it must maintain a cache of connections.	Page MS-SUB EPIC000008 states that the Username field in a Web Database Gateway (WDG) file "must be a valid login to a SQL Server." SQL Server is a server by which a database (data source) is accessed.	"By default, Httpodbc.dll caches the Web page sent to the client. If a subsequent request is identical, the cached page will be returned without ever accessing the database." MS-SUB EPIC 000015	The reference expressly teaches HTML extension (.HTX) files. "The HTX file is the template for the actual HTML document that will be returned to the Web browser after the database information has been merged into it by Httpodbc.dll." MS-SUB EPIC 000005. "The [HTX] file is an HTML document that contains placeholders for data returned from the database." MS-SUB EPIC 000010.	The insertion is taught at page MS-SUB EPIC 000010: "After the SQL statement has been executed, Httpodbc.dll reads the HTX file … and replaces every placeholder with the actual data returned by the query."	Microsoft IIS is a network-based systems for handling dynamic Web page generation requests	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098). A processor of a computer that runs Httpodbc.dll is the page server processing means
38				The reference teaches custom templates for configuring reports (web pages): "Smart Report Definitions are System Templates that have been customized to include particular Business Concepts, Business Indicators, and/or segments. Smart Report Definitions can be immediately "run" to produce a "Smart Report", saved to be run later or saved and scheduled to be run later." 4:44-48. "Smart Report viewing subsystem 53 includes a parser which parses the Smart Report, which is in an HTML file." 9:15-17	"Smart Report generator 72 fulfills the primary purpose of DAI subsystem 14. Report generation begins when a user chooses a Smart Report and an appropriate set of parameters in client subsystem 12. This Smart Report Request is used to first fetch a System Template. A System Template consists of a set of Abstract Queries. An Abstract Query contains placeholders for Business Concepts and Business Indicators that are filled in with the user selections in the process of instantiation, as well as conditions that are evaluated to generate different output and actual layout information. The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML)" 14:31-45	The system is networked. See Fig. 1	See '554 1d1	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
37	The first computer system includes a computer that has a processor that runs a web browser as shown on page MS-SUB EPIC000097.	The second computer system includes a processor that runs the MS IIS as shown MS-SUB EPIC000097	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to “get or post information in an ODBC database” (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098). The means is a processor of a computer that runs MS IIS. See figure on MS-SUB EPIC000097	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to “get or post information in an ODBC database” (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098). The router is a component of the computer running the MS-IIS (such as the WWW Service component). See figure on MS-SUB EPIC000098. This figure shows the WWW Service component routing the request from MS IIS to Httpodbc.dll.	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between “WWW service” and “Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request (“intercepting” it) and hands it off to the page server. (See MS-SUB EPIC 000007-0000010; 000097-000098).	The dispatcher is that component of MS IIS that makes an informed decision to forward the request to Httpodbc.dll, as opposed to other services.	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. “Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs.” (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed
38	The first computer system includes the client subsystem 12 of Fig. 1	The second computer system includes the server computer 32 of Fig. 1	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	The page server must receive the request to process it. Receipt is shown in Fig. 1.

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
37	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	This function is performed by the processor of the computer running httpodbc.dll. See also '554 1d2.	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.	Microsoft IIS is a network-based systems for handling dynamic Web page generation requests	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between "WWW service" and "Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request ("intercepting" it) and hands it off to the page server. (See MS-SUB EPIC 000007 0000010; 000097-000098).	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. "Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs." (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	IIS is a computer system, whose instructions are contained on a machine readable medium	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed
38	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:40-45	See '554 1d1	The system is networked. See Fig. 1	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	Because the disclosed system is implemented on computers, the instructions must be on a machine-readable medium	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The page server must receive the request to process it. Receipt is shown in Fig. 1.

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
37	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between “WWW service” and “Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request (“intercepting” it) and hands it off to the page server. (See MS-SUB EPIC 000007-0000010; 000097-000098).	The dispatcher is that component of the Database Connector which reads the WDG file. The URL of a request will reference the Database Connector (Httpodbc.dll) as well as a “WDG file” that “contain[s] the necessary information to connect to the desired ODBC data source …” (See MS-SUB EPIC 000005; 000098) Using this information, a component of the Web server of the MS IIS invokes the Database Connector and passes the request parameters to it.	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. “Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs.” (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	“Httpodbc.dll obtains the name of the WDG file [specified in the request]. . .Httpodbc.dll reads the WDG file . . . Httpodbc.dll connects to the ODBC data source, and executes the SQL statement contained within the DG file . . .” (MS-SUB EPIC 000009). IIS supports multiple simultaneous (concurrent) connections. (See MS-SUB EPIC 000071-000072).	This function is performed by httpodbc.dll. See also '554 1d2.	'335	“Your Internet Server can contain anything from static pages of information to interactive programs that take data and return customized pages for remote users." MS-SUB EPIC000092. “[w]ith the Web Server and ODBC, you can create Web pages with information contained in a database,” thus enabling generation of dynamic Web pages. (MS-SUB EPIC 000004, 000097)	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to “get or post information in an ODBC database” (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed
38	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind-Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	"Thus, system 10 is implemented as a three-tier architecture." 5:48-49. Concurrency is inherent in a three-tier architecture. "Server computer 32 is preferably a multi-processor computer" 5:64-65. Concurrency is inherent in a multiprocessor architecture	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:40-45	'335	The reference discloses a Web system for generating dynamic documents: "An Abstract Query contains placeholders for Business Concepts and Business Indicators that are filled in with the user selections in the process of instantiation, as well as conditions that are evaluated to generate different output and actual layout information. The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:36-45	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The page server must receive the request to process it. Receipt is shown in Fig. 1.

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
37	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between "WWW service" and "Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request ("intercepting" it) and hands it off to the page server. (See MS-SUB EPIC 000007-0000010; 000097-000098).	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).	"Httpodbc.dll obtains the name of the WDG file [specified in the request]. . .Httpodbc.dll reads the WDG file . . . Httpodbc.dll connects to the ODBC data source, and executes the SQL statement contained within the DG file . . ." (MS-SUB EPIC 000009). IIS supports multiple simultaneous (concurrent) connections. (See MS-SUB EPIC 000071-000072).	This function is performed by httpodbc.dll. See also '554 1d2.	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.	"Your Internet Server can contain anything from static pages of information to interactive programs that take data and return customized pages for remote users." MS-SUB EPIC000092. "[w]ith the Web Server and ODBC, you can create Web pages with information contained in a database," thus enabling generation of dynamic Web pages. (MS-SUB EPIC 000004, 000097)	The dispatcher is that component of the Database Connector which reads the WDG file. The URL of a request will reference the Database Connector (Httpodbc.dll) as well as a "WDG file" that "contain[s] the necessary information to connect to the desired ODBC data source …" (See MS-SUB EPIC 000005; 000098) Using this information, a component of the Web server of the MS IIS invokes the Database Connector and passes the request parameters to it.
38	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind-Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	"Thus, system 10 is implemented as a three-tier architecture." 5:48-49. Concurrency is inherent in a three-tier architecture. "Server computer 32 is preferably a multi-processor computer" 5:64-65. Concurrency is inherent in a multiprocessor architecture	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely used standard for building WYSIWYG compound documents." 14:40-45	See '554 1d1	The reference discloses a Web system for generating dynamic documents: "An Abstract Query contains placeholders for Business Concepts and Business Indicators that are filled in with the user selections in the process of instantiation, as well as conditions that are evaluated to generate different output and actual layout information. The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:36-45	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
37	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. "Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs." (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	The WDG file referenced in the URL of a request for a dynamic Web page includes information identifying the "data source" from which data is to be retrieved. See the sample file on page MS-SUB EPIC 000008	The page server (httpodbc.dll) "connects to the ODBC data source" and executes SQL statements against the data source to obtain data for the dynamic web page. MS-SUB EPIC 000009	The diagram on page MS-SUB EPIC000098 shows that ODBC can connect simultaneously to multiple databases. (It must handle concurrent requests which require different databases.) For these simultaneous connections it must maintain a cache of connections.	Page MS-SUB EPIC000008 states that the Username field in a Web Database Gateway (WDG) file "must be a valid login to a SQL Server." SQL Server is a server by which a database (data source) is accessed.	"By default, Httpodbc.dll caches the Web page sent to the client. If a subsequent request is identical, the cached page will be returned without ever accessing the database." MS-SUB EPIC 000015	HTML extension templates are tag-based templates for configuring a dynamic page. See '554 7	Insertion into HTML extension templates is insertion into tag-based templates. See '554 8	An HTX file specifies the layout of the returned Web page and places where the retrieved data will appear at on that Web page. One of the functions of HTML is to specify format
38	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	"When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14." 5:41-43. The required identification is based on the list of Alerts and Scheduled Smart Report Requests	"DSM subsystem 16 reads schema from data warehouse 24, creates data views, and creates a mapping between the two. It also uses that mapping to translate the Dimensional Queries received from DAI subsystem 14 into SQL and package and return the results." 5:32-36. The data is retrieved via an SQL query				The Smart Reports are tag-based (HTML) templates	See '554 8	"Smart Report viewing subsystem 53 includes a parser which parses the Smart Report, which is in an HTML file. In an HTML file, HTML tags indicate document elements, structure, formatting, and hypertext linking to other documents or to included media. The parser then outputs all the information for display." 9:15-20. The purpose of the HTML is to drive a format of the data

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
37	HTX files without extensions are HTML templates	"Update" is a standard SQL command. "With the WWW service and the ODBC drivers provided with Microsoft Internet Information Server, you can: Create Web pages with information contained in a database. Insert, update, and delete information in the database based on user input from a Web page…" MS-SUB EPIC 000091.			IIS is computer-implemented	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	A dynamic page generation request is intercepted at the web server and sent to httpodbc.dll for processing. This is shown in the diagram on p. 52 as the arrow between "WWW service" and "Httpodbc.dll." If the request is for a dynamic page, the WWW Service interrupts its processing of the request ("intercepting" it) and hands it off to the page server. (See MS-SUB EPIC 000007-0000010; 000097-000098).	The page server is the "Database Connector httpodbc.dll" of the reference, which applications use to "get or post information in an ODBC database" (MS-SUB EPIC000096)." The double arrows on the diagram on page 52 (EPIC000098) show request routing. Httpodbc.dll fetches requested data from a SQL Server database and returns a dynamically generated Web page. (MS-SUB EPIC 000007-0000010; 000097-000098).
38	See '335 8	The invention uses SQL, one of whose primitives is UPDATE. Updating is also taught expressly: "This metadata 25 represents Business Concepts and Business Indicators that provide a customizable "dimensionalization" of the relational data in data warehouse 24. DAI subsystem 14 also processes updates to this metadata 25 that originate in client subsystem 12 and handles several other kinds of user updates, primarily by passing them to DSM subsystem 16." 5:25032	See '335 14	OLE is taught expressly: "Object Linking and Embedding (OLE) is a computer format that allows objects (e.g., graphs, tables) in computer documents to, when double clicked on, bring up the software application that created the object (graph, table, document). ... Reports are compound documents that display data from a database in text and graphics (e.g., graphs, tables). Reports are the result of running a Smart Report Definition. Smart Reports are in the HTML format and are OLE 2.0 compliant." 4:27-43 "	The disclosed system is computer implemented	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1	The page server must receive the request to process it. Receipt is shown in Fig. 1.	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind-Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	The request is intercepted because it is not handled by the Web server but is passed off to the page server: "[S]ubsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:44-47	The Web server is "server computer 32." The page server is "DSM subsystem 16" and that component of "DAI Subsystem 14" which fills in HTML templates. The routing is shown in Fig. 1

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
37	"Httpodbc.dll obtains the name of the WDG file [specified in the request]. . .Httpodbc.dll reads the WDG file . . . Httpodbc.dll connects to the ODBC data source, and executes the SQL statement contained within the DG file . . ." (MS-SUB EPIC 000009). IIS supports multiple simultaneous (concurrent) connections. (See MS-SUB EPIC 000071-000072).	This function is performed by httpodbc.dll. See also '554 1d2.	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.	IIS is computer-implemented	The dispatcher is that component of the Database Connector which reads the WDG file. The URL of a request will reference the Database Connector (Httpodbc.dll) as well as a "WDG file" that "contain[s] the necessary information to connect to the desired ODBC data source …" (See MS-SUB EPIC 000005; 000098) Using this information, a component of the Web server of the MS IIS invokes the Database Connector and passes the request parameters to it.	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. "Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs." (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	The WDG file referenced in the URL of a request for a dynamic Web page includes information identifying the "data source" from which data is to be retrieved. See the sample file on page MS-SUB EPIC 000008	The page server (httpodbc.dll) "connects to the ODBC data source" and executes SQL statements against the data source to obtain data for the dynamic web page. MS-SUB EPIC 000009	The diagram on page MS-SUB EPIC000098 shows that ODBC can connect simultaneously to multiple databases. (It must handle concurrent requests which require different databases.) For these simultaneous connections it must maintain a cache of connections.
38	"Thus, system 10 is implemented as a three-tier architecture." 5:48-49. Concurrency is inherent in a three-tier architecture. "Server computer 32 is preferably a multi-processor computer" 5:64-65. Concurrency is inherent in a multiprocessor architecture	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:40-45	See '554 1d1	The disclosed system is computer implemented	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	"When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14." 5:41-43. The required identification is based on the list of Alerts and Scheduled Smart Report Requests	"DSM subsystem 16 reads schema from data warehouse 24, creates data views, and creates a mapping between the two. It also uses that mapping to translate the Dimensional Queries received from DAI subsystem 14 into SQL and package and return the results." 5:32-36. The data is retrieved via an SQL query	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
37	Page MS-SUB EPIC000008 states that the Username field in a Web Database Gateway (WDG) file "must be a valid login to a SQL Server." SQL Server is a server by which a database (data source) is accessed.	"By default, Httpodbc.dll caches the Web page sent to the client. If a subsequent request is identical, the cached page will be returned without ever accessing the database." MS-SUB EPIC 000015	HTML extension templates are tag-based templates for configuring a dynamic page. See '554 7	Insertion into HTML extension templates is insertion into tag-based templates. See '554 8	An HTX file specifies the layout of the returned Web page and places where the retrieved data will appear at on that Web page. One of the functions of HTML is to specify format	HTX files without extensions are HTML templates	"Update" is a standard SQL command. "With the WWW service and the ODBC drivers provided with Microsoft Internet Information Server, you can: Create Web pages with information contained in a database. Insert, update, and delete information in the database based on user input from a Web page…" MS-SUB EPIC 000091.			IIS is computer-implemented
38			The Smart Reports are tag-based (HTML) templates	See '554 8	"Smart Report viewing subsystem 53 includes a parser which parses the Smart Report, which is in an HTML file. In an HTML file, HTML tags indicate document elements, structure, formatting, and hypertext linking to other documents or to included media. The parser then outputs all the information for display." 9:15-20. The purpose of the HTML is to drive a format of the data	See '335 8	The invention uses SQL, one of whose primitives is UPDATE. Updating is also taught expressly: "This metadata 25 represents Business Concepts and Business Indicators that provide a customizable "dimensionalization" of the relational data in data warehouse 24. DAI subsystem 14 also processes updates to this metadata 25 that originate in client subsystem 12 and handles several other kinds of user updates, primarily by passing them to DSM subsystem 16." 5:25032	See '335 14	OLE is taught expressly: "Object Linking and Embedding (OLE) is a computer format that allows objects (e.g., graphs, tables) in computer documents to, when double clicked on, bring up the software application that created the object (graph, table, document). ... Reports are compound documents that display data from a database in text and graphics (e.g., graphs, tables). Reports are the result of running a Smart Report Definition. Smart Reports are in the HTML format and are OLE 2.0 compliant." 4:27-43 "	The disclosed system is computer implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
37	The dispatcher is that component of the Database Connector which reads the WDG file. The URL of a request will reference the Database Connector (Httpodbc.dll) as well as a "WDG file" that "contain[s] the necessary information to connect to the desired ODBC data source …" (See MS-SUB EPIC 000005; 000098) Using this information, a component of the Web server of the MS IIS invokes the Database Connector and passes the request parameters to it.	The WDG file reference in the URL of the request for a dynamic Web page includes information identifying the "data source" from which data is to be retrieved. WDG files can be updated at any time. More important, the WDG file is interrogated dynamically when a request is received. The figure at MS-SUB EPIC000098 shows Httpodbc.dll having access to multiple data sources. Also, figure 18 at MS-SUB EPIC000022 shows that it is possible to install multiple data sources. To retrieve data residing in one or more data sources, maintaining at least some information about the data sources is required. The WDG file specifies information about data sources. See MS-SUB EPIC004015. "Datasource[:] The name that corresponds to the data source name you created earlier by using the ODBC Administrator." Id.	The selection of page server made by the dispatcher is "informed" because it is based on parameters contained in the request and information in a WDG file (p. 3, MS-SUB EPIC 000006), which may be modified dynamically. This is configuration file dispatching, according to EpicRealm's contentions. The Internet Database Connector is one of possibly many Internet Server API applications that are capable of processing the request. "Access to databases is accomplished through a component of Internet Information Server called the Internet Database Connector, one of a family of Internet Server APIs." (MS-SUB EPIC 000097). As mentioned above, the Database Connector functions as a Page Server to process the request for dynamic content. Thus, the component of the Web server that hands off the request to the Internet Database Connectors is a dispatcher that dispatches the request to a page server – the Database Connector (Httpodbc.dll). (See MS-SUB EPIC 000007-0000010; 000097-000098).	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	"Httpodbc.dll obtains the name of the WDG file [specified in the request]. . .Httpodbc.dll reads the WDG file . . . Httpodbc.dll connects to the ODBC data source, and executes the SQL statement contained within the DG file . . ." (MS-SUB EPIC 000009). IIS supports multiple simultaneous (concurrent) connections. (See MS-SUB EPIC 000071-000072).	This function is performed by httpodbc.dll. See also '554 1d2.	The dynamically generated web page contains data accessed from databases. MS-SUB EPIC000005. "After the SQL statement has been executed, Httpodbc.dll reads the HTX file specified in Sample.wdg (Sample.htx) and replaces every placeholder with the actual data returned by the query." MS-SUB EPIC 000010.
38	The dispatcher is "Scheduler subsystem 18." "Scheduler subsystem 18 is responsible for launching Alerts and running any Smart Reports which the user specified to execute in batch mode at some specified time interval. When the requested time interval occurs, the Scheduler starts up, requests a list of Alerts and scheduled Smart Report Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12." 5:37-47.	This information is the metadata 25: "DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data. DAI subsystem 14 is responsible for instantiating user selected Smart Reports and managing several kinds of metadata 25 used in this instantiation. This metadata 25 represents Business Concepts and Business Indicators that provide a customizable "dimensionalization" of the relational data in data warehouse 24. DAI subsystem 14 also processes updates to this metadata 25 that originate in client subsystem 12 and handles several other kinds of user updates, primarily by passing them to DSM subsystem 16." 5:17-31	The Scheduler 18 makes an informed selection of which page server should be invoked to process the request. Thus it dispatches. See '554 1b5. "The user also specifies how often to check the Alert condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This metadata 25 is also available to DAI subsystem 14." 10:67-11:4. The dynamic information maintained about page servers is which have signalled Alerts	The page server must receive the request to process it. Receipt is shown in Fig. 1.	"Server computer 32 is preferably a multi-processor computer and executes the UNIX operating system or Windows NT, although other computer and operating system configurations are also envisioned by the present invention." 5:64-67. Interprocess communication under Unix is via Socket-Bind-Listen, so the web server is released via SBL releasing, according to EpicRealm's contention, when it passes off the page request to the page server	"Thus, system 10 is implemented as a three-tier architecture." 5:48-49. Concurrency is inherent in a three-tier architecture. "Server computer 32 is preferably a multi-processor computer" 5:64-65. Concurrency is inherent in a multiprocessor architecture	"DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data." 5:17-23. The "text" comprises web pages: "The output of the Report Generation process is a fully instantiated Smart Report returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building WYSIWYG compound documents." 14:40-45	See '554 1d1

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
39	002	Varela et al., "DB: Browsing Object-Oriented Databases Over the Web"	Dec. 1995		'554	DB is a computer system for generating dynamic pages: "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers." p. 3	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.
40	00Q	Antchev, "A WWW Learning Environment for Mathematics"	Dec. 1995		'554	The system generates dynamic pages in response to requests to an HTTP (Web) server: "The handler being invoked should return the appropriate HTML or other type of document or it should generate such a document on the fly. Our CGI-handlers fall into the second category--an HTML document is generated and returned to the HTTP server." p. 4	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES	The request must be received to be processed	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5). This is TCP releasing according to EpicRealm's contentions	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
39	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.	The "same server can also be used to handle operations on multiple databases." The choice is not arbitrary, but is determined from the query, which includes "database_name"	See '554 1d2.
40	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	The reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4. "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5	"The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings." (p.5, para. 6 under "Implementation.). The HTTP request which spawns the request to Mathematica includes a CGI request that "identifies" the Mathematica data source and context. p. 5. Further, the HTML templates provided to Exercise Server include strings generated by the CGI-Handler that identify the sources of interactive elements to be included in the generated HTML document. p. 5.	"These strings [identifying dynamically retrieved elements] are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." (p 5, para. 6 under "Implementation.).

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
39	"[G]iven the statelessness of HTTP, we introduced a dispatcher script architecture. In this architecture, a CGI script communicates with an intermediate data server connected to the database application, which keeps the database open for faster future transactions." Abstract. "The main characteristic of the intermediate data server is that it remains alive across multiple requests and thereby eliminates the earlier problem of the script ceasing to exist after every operation." p. 6. The set of active connections is a connection cache		The page-caching improves performance by eliminating some system overhead: "[w]e also observed that closing the database does not actually result in a subsequent request taking a 'cold time.' This result, we concluded was because Objectstore has a page server model which means that even when the database is closed, the data pages are still in memory and are not reclaimed unless the data of some other application is paged into the same place." p. 10. This is page caching. The generated pages are stored in memory and, if needed, do not have to be generated anew.	"When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers. For such instances, there are scripts, which are programs that perform conversions from different data formats into HTML on-the-fly." p. 3. The dispatcher script "provides representation of the object contents in HTML form. This part not only displays data, but also creates hyperlinks to related objects, superclasses, etc., in the same format as in the actual queries." p. 9. This representation is a custom HTML extension template	The dispatcher script "makes a connection with the database server, sends the query, and accepts the result stream from the server for utilization by other routines." p. 9. This is the dynamically retrieved data that is inserted into the template	The reference describes an implementation of object-oriented databases on the World-Wide Web, and thus discloses a networked-system because the Web is a network of computers. Its purpose is to handle dynamic web page generation requests	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server. The processor of a computer that runs the Intermediate data server is the page server processing means.
40	The Exercise Server (page server) maintains state information about the different Mathematica contexts in the Exercise Server's main memory: This "simplifies the task of maintaining the state information. It is no longer necessary to encode/decode data structures to/from flat representation at each HTTP request because it can be kept throughout the entire session in Exercise Server's main memory." p. 4. This storage of the state of connections to Mathematica contexts is a connection cache			HTML extension templates are expressly taught. "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server. The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings." p. 5.	"These strings [from the HTML templates] are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." p. 5	The reference teaches a networked system. "The communication between the CGI-handlers and ES is based on Berkeley Sockets API … and allows ES to be run either on the same computer, where the HTTP-server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES. The processing means is the processor of a computer that is running the Exercise Server software.

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
39	Figure 2 shows the WWW client. MAC000050. The first computer is a computer that has a processor that runs a browser of the WWW client as shown on MAC000050.	Second computer system includes a computer running the WWW server.	The Web server is "WWW Server" of Fig. 2. The means for receiving is a processor of a computer that runs the WWW server.	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.
40	The first computer system is the "standard HTTP client." p. 4. The "means for generating said request" is the processor of the computer running a standard HTTP client. See MAC004603.	The second computer system includes the HTTP sever, the CGI-handler and the Exercise Servers.	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES. The "means for receiving said request" is the processor of a computer running an HTTP server.	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES. The processor of the computer running the Exercise Server (page server) software "replies to any request by generating a HTML-document." p. 5	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
39	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.	The reference describes an implementation of object-oriented databases on the World-Wide Web, and thus discloses a networked-system because the Web is a network of computers. Its purpose is to handle dynamic web page generation requests	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	Because the disclosed system is implemented on computers, the instructions must be on a machine-readable medium	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.
40	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5). This is TCP releasing according to EpicRealm's contentions	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5	The reference teaches a networked system. "The communication between the CGI-handlers and ES is based on Berkeley Sockets API … and allows ES to be run either on the same computer, where the HTTP-server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	Because the disclosed system is implemented on computers, the instructions must be on a machine-readable medium	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
39	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	'335	DB is a computer system for generating dynamic pages: "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers." p. 3	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.
40	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5). This is TCP releasing according to EpicRealm's contentions	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	The reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4. "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	'335	The system generates dynamic pages in response to requests to an HTTP (Web) server: "The handler being invoked should return the appropriate HTML or other type of document or it should generate such a document on the fly. Our CGI-handlers fall into the second category--an HTML document is generated and returned to the HTTP server." p. 4	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
39	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the "Dispatcher script" and the "Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.	DB is a computer system for generating dynamic pages: "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers." p. 3	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.
40	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5). This is TCP releasing according to EpicRealm's contentions	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). "ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES	The reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4. "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5	The system generates dynamic pages in response to requests to an HTTP (Web) server: "The handler being invoked should return the appropriate HTML or other type of document or it should generate such a document on the fly. Our CGI-handlers fall into the second category--an HTML document is generated and returned to the HTTP server." p. 4	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
39	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	The "same server can also be used to handle operations on multiple databases." The choice is not arbitrary, but is determined from the query, which includes "database_name"	See '554 1d2.	"[G]iven the statelessness of HTTP, we introduced a dispatcher script architecture. In this architecture, a CGI script communicates with an intermediate data server connected to the database application, which keeps the database open for faster future transactions." Abstract. "The main characteristic of the intermediate data server is that it remains alive across multiple requests and thereby eliminates the earlier problem of the script ceasing to exist after every operation." p. 6. The set of active connections is a connection cache		The page-caching improves performance by eliminating some system overhead: "[w]e also observed that closing the database does not actually result in a subsequent request taking a 'cold time.' This result, we concluded was because Objectstore has a page server model which means that even when the database is closed, the data pages are still in memory and are not reclaimed unless the data of some other application is paged into the same place." p. 10. This is page caching. The generated pages are stored in memory and, if needed, do not have to be generated anew.	See '554 7	See '554 8.	The templates drive a format of the data. "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers. For such instances, there are scripts, which are programs that perform conversions from different data formats into HTML on-the-fly." p. 3
40	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	"The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings." (p.5, para. 6 under "Implementation.). The HTTP request which spawns the request to Mathematica includes a CGI request that "identifies" the Mathematica data source and context. p. 5. Further, the HTML templates provided to Exercise Server include strings generated by the CGI-Handler that identify the sources of interactive elements to be included in the generated HTML document. p. 5.	"These strings [identifying dynamically retrieved elements] are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." (p 5, para. 6 under "Implementation.).	The Exercise Server (page server) maintains state information about the different Mathematica contexts in the Exercise Server's main memory: This "simplifies the task of maintaining the state information. It is no longer necessary to encode/decode data structures to/from flat representation at each HTTP request because it can be kept throughout the entire session in Exercise Server's main memory." p. 4. This storage of the state of connections to Mathematica contexts is a connection cache			The tag-based text templates are the HTML extension templates. See '554 7.	See '554 8.	The purpose of the HTML templates is to drive the format of the data

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
39	The templates are HTML-based		"we used ODL, a standard object-definition language, and defined a simple intermediate data format for moving database objects across the network." Abstract. ODL is an extension that permits object handling		The disclosed system is computer implemented	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the “Dispatcher script” and the “Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	Instead of processing the request on its own, as seen in the figure on MAC000050, the WWW server passes it to the dispatcher script.	The Web server is "WWW Server" of Fig. 2. The page server is the combination of the “Dispatcher script” and the “Intermediate data server" in Fig. 2, which shows the routing. The reference refers to a "page server architecture." p. 2. Alternatively, the Intermediate data server is a page server.
40	The tag-based text templates are HTML templates. See '335 8.				The disclosed system is computer implemented	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). “ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES	The request must be received to be processed	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network.” (p. 5). This is TCP releasing according to EpicRealm's contentions	The request is not handled by the HTTP (web) server, but is intercepted prior to handling and sent to a newly-spawned CGI-handler. Its handling is thus intercepted	The Web server is the HTTP server described on p. 5. The page server is the Exercise Server (ES). “ES replies to any request by generating a HTML-document. It is based on a HTML-template which holds the static parts of the document--those which do not depend on the interaction between the user and the server." p. 5. The request is routed from the HTTP server to the ES

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
39	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.	The disclosed system is computer implemented	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	The "same server can also be used to handle operations on multiple databases." The choice is not arbitrary, but is determined from the query, which includes "database_name"	See '554 1d2.	"[G]iven the statelessness of HTTP, we introduced a dispatcher script architecture. In this architecture, a CGI script communicates with an intermediate data server connected to the database application, which keeps the database open for faster future transactions." Abstract. "The main characteristic of the intermediate data server is that it remains alive across multiple requests and thereby eliminates the earlier problem of the script ceasing to exist after every operation." p. 6. The set of active connections is a connection cache
40	The reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4. "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5	The disclosed system is computer implemented	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	"The HTML-templates are ordinary HTML files which are normally created with an HTML-editor or any text editor. The places, where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings." (p.5, para. 6 under "Implementation."). The HTTP request which spawns the request to Mathematica includes a CGI request that "identifies" the Mathematica data source and context. p. 5. Further, the HTML templates provided to Exercise Server include strings generated by the CGI-Handler that identify the sources of interactive elements to be included in the generated HTML document. p. 5.	"These strings [identifying dynamically retrieved elements] are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents, which are sent to the clients." (p 5, para. 6 under "Implementation.).	The Exercise Server (page server) maintains state information about the different Mathematica contexts in the Exercise Server's main memory: This "simplifies the task of maintaining the state information. It is no longer necessary to encode/decode data structures to/from flat representation at each HTTP request because it can be kept throughout the entire session in Exercise Server's main memory." p. 4. This storage of the state of connections to Mathematica contexts is a connection cache

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
39		The page-caching improves performance by eliminating some system overhead: “[w]e also observed that closing the database does not actually result in a subsequent request taking a ‘cold time.’ This result, we concluded was because Objectstore has a page server model which means that even when the database is closed, the data pages are still in memory and are not reclaimed unless the data of some other application is paged into the same place.” p. 10. This is page caching. The generated pages are stored in memory and, if needed, do not have to be generated anew.	See '554 7	See '554 8.	The templates drive a format of the data. "When the documents to be published are dynamic, such as those resulting from queries to databases, the hypertext needs to be generated by the servers. For such instances, there are scripts, which are programs that perform conversions from different data formats into HTML on-the-fly." p. 3	The templates are HTML-based		"we used ODL, a standard object-definition language, and defined a simple intermediate data format for moving database objects across the network." Abstract. ODL is an extension that permits object handling		The disclosed system is computer implemented
40			The tag-based text templates are the HTML extension templates. See '554 7.	See '554 8.	The purpose of the HTML templates is to drive the format of the data	The tag-based text templates are HTML templates. See ‘335 8.				The disclosed system is computer implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
39	The Intermediate data server receives the request. "Whenever the dispatcher script gets a request, it connects to the data server which is listening for requests and sends the request in our own simple intermediate data exchange format described in the following section." MAC000051.	The reference teaches a server that handles operations on multiple databases through the use of lookup tables. "The same server can also be used to handle operations on multiple databases. This is achieved through liberal use of lookup tables to perform name-to-object and name-to-function translations." p. 6. The disclosure of lookup tables to manage handling of different databases inherently discloses "dynamic information regarding data sources" because the date in the lookup tables can change (is dynamic).	The Dispatcher Script includes a Request Manager which "determines the type of query and references the appropriate routine." "The dispatcher program is a CGI script which provides a user-friendly interface to query the intermediate data server and returns the database schema, extent, and object data to the WWW browser. Separating this dispatcher from the intermediate data server is beneficial for several reasons. First, the separation allows us to keep the intermediate data server active even when the transaction is over." p. 6. This inherently describes the ability to route a request to an intermediate data server based on the database resources available to that server. See also MAC000057 (architecture contemplates specialized servers such as video servers, annotation servers, …). It therefore makes an informed selection which page server to invoke based on information about page servers that can change dynamically. See also '335 29c.	The dispatcher script and intermediate data server (page server) must receive the request to process it. Alternatively, the Intermediate data server receives the request as seen in the figure at MAC000050.	The web server hands off the request to the Intermediate data server using TCP, as shown in Fig. 2. TCP provides acknowledgment of receipt, which is form of releasing, according to EpicRealm. Also, the "releasing" limitation is satisfied under the SBL theory because the Intermediate data server has to bind to a TCP port.	That the web server may process requests concurrently while the page server performs its processing is inherent in the architecture show in the diagram. This is the reason that the dynamic request is handed off to a page server. Otherwise, the web server would be capable of handling the request itself, but does not do so to keep it free to process other requests	Dynamic documents "like those resulting from queries to databases" are created ("converted into HTML") "on-the-fly." "The [Intermediate] data server performs the database operation and sends back the results using the same connection to the dispatcher script." MAC000051. The results are data displayable by a web browser. ". . . [W]e have only used data in text or numerical format." MAC000052. See also MAC000052-53.	The Web Page includes data dynamically retrieved from the Objectstore object-oriented database management system on the database server and storage shown at MAC000050.
40	The CGI-handler functions as dispatcher. The request is routed from the HTTP (web) server) to the CGI-handler	The Exercise Server (page server) can access different Mathematica "contexts" (MAC004603-4604) and maintains dynamic information regarding the different Mathematica "contexts" corresponding to concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4	The CGI-Handler (dispatcher) performs Single-Server Dispatching. It examines the request and extracts the "context" variable from a hidden field of the request form and including the "context" variable in the DoSubmitForm command. The "context" variable in the DoSubmitForm command uniquely identifies one of the concurrent Mathematica sessions to be accessed by an ES (page server). Under an alternative theory, the ES combined with a Mathematica session is a "page server." The "concurrent Mathematica sessions" are being started and terminated dynamically, so the CGI-handler performs dispatching and bases its selection on dynamic information maintained about page servers	The request must be received to be processed	"Upon the client's request, the HTTP server spawns the corresponding CGI-handler and passes the encoded form's content to it (assuming the request is using the POST method; see Figure 1). The later [sic] decodes the form's content and connects to ES. The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." (p. 5). This is TCP releasing according to EpicRealm's contentions	The reference expressly teaches concurrent sessions: "Contexts in Mathematica facilitate distinguishing different uses of a particular name and, as such, are used by our interactive exercises to store the state information of concurrent sessions." p. 4. "Upon the client's request, the HTTP server spawns the corresponding CGI-Handler … The communication between the CGI-handlers and ES is based on Berkeley Sockets API [11] and allows ES to be run either on the same computer, where the HTTP server and CGI-handlers are running, or on a remote one over a TCP/IP network." p. 5. At least in the case of a remote ES, the HTTP server continues to process requests.	The Exercise Server (page server) "replies to any request by generating a HTML-document." p. 5	The page generated by the Exercise Server includes content dynamically retrieved from different Mathematica contexts (data sources). The places in the HTML templates "where interactive elements (formulas, plots, text responses, etc.) should appear, are marked with designated strings. These strings are then replaced by ES with relevant HTML-elements in order to produce the final HTML-documents." p. 5

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
41	039	Gaines, "Porting Interactive Applications to the Web"	Dec. 1995		'554	Fig. 1 shows a computer-implemented method for "client-server transactions responding by generating dynamic documents"	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the "server." The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The request must be received to be processed	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.
42	00O	Ingham et al, "W3Objects: Bringing Object-Oriented Technology to the Web," 4th WWW Conf.	Dec. 1995		'554	The reference teaches accessing WWW objects dynamically and displaying their data on Web browsers	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The page server receives the request in Fig. 5	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
41			Because HTTP (the web server) is released per '554 1b3, it is free to process other requests	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as “databases” and “computational services.”	The server gateway must identify which of the “databases” and “computational services it will access	See '554 1d2
42	The Web server sends the request to the HTTP daemon (dispatcher)	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	Concurrency is inherent in the architecture of Fig. 5. When a request is passed off to a page server, the web server can continue to accept incoming requests	Fig. 5 shows the W3Object Server responding to the request	The objects accessed by the W3Object Server are the data sources	See '554 1b6	The retrieval occurs when the referenced object is accessed

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
41				The customer HTML extension template is the "representation of the application output as a document," which receives data form data sources to generate the dynamic page. The template must be an extension to accommodate substitution of markers by dynamic data. HTML is disclosed in the reference. See Sec. 4.3, which discusses "user customization"	This is done via WebGrid scripts: "In the script text a backslash followed by a number indicates a parameter to be substituted by some data already set up in the code. A backslash followed by a "#" and a number introduces a list of possible strings to be selected on the basis of a numeric parameter set up by the code." Sec. 6.	Fig. 1 shows a networked system for "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as "databases" and "computational services."	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the "server." The routing is shown by arrows between "HTTP Protocol" and "Encoding"
42	The HTTP Gateway inherently maintains a connection cache because it maintains simultaneous connections to multiple W3Object Servers, as shown in Fig. 5					The disclosed system is Web-based, hence networked. See '554 1	The objects accessed by the W3Object Server are the data sources	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
41	The first computer system is shown as "Client" in Fig. 2	The second computer system is shown as "HTTP" and "SERVER" in Fig. 2	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the “server.” The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the “server.” The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.			The request must be received to be processed
42	The first computer system includes the web browser of Fig. 5	The second computer system includes the HTTP Gateway and W3Object Servers of Fig. 5	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor	The Web server sends the request to the HTTP daemon (dispatcher)	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	The page server receives the request in Fig. 5

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
41	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as “databases” and “computational services.”	Fig. 1 shows a networked system for "client-server transactions responding by generating dynamic documents"	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.		Because the system is a programmed system, it must have a medium on which instructions are stored.	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the “server.” The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The request must be received to be processed
42	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	Fig. 5 shows the W3Object Server responding to the request	The objects accessed by the W3Object Server are the data sources	The disclosed system is Web-based, hence networked. See '554 1	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	Because the system is a programmed system, it must have a medium on which instructions are stored.	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The page server receives the request in Fig. 5

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
41	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.			Because HTTP (the web server) is released per '554 1b3, it is free to process other requests	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	'335	Fig. 1 shows a computer-implemented method for "client-server transactions responding by generating dynamic documents"	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the "server." The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The request must be received to be processed
42	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor	The Web server sends the request to the HTTP daemon (dispatcher)	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	Concurrency is inherent in the architecture of Fig. 5. When a request is passed off to a page server, the web server can continue to accept incoming requests	Fig. 5 shows the W3Object Server responding to the request	'335	The reference teaches accessing WWW objects dynamically and displaying their data on Web browsers	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The page server receives the request in Fig. 5

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
41	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the "server." The routing is shown by arrows between "HTTP Protocol" and "Encoding"	Because HTTP (the web server) is released per '554 1b3, it is free to process other requests	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as "databases" and "computational services."	Fig. 1 shows a computer-implemented method for "client-server transactions responding by generating dynamic documents"	
42	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	Concurrency is inherent in the architecture of Fig. 5. When a request is passed off to a page server, the web server can continue to accept incoming requests	Fig. 5 shows the W3Object Server responding to the request	The objects accessed by the W3Object Server are the data sources	The reference teaches accessing WWW objects dynamically and displaying their data on Web browsers	The Web server sends the request to the HTTP daemon (dispatcher)

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
41		The server gateway must identify which of the “databases” and “computational services it will access	See '554 1d2				See '554 7	See '554 8.	The purpose of HTML in the templates is to derive a format of the data
42	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	See '554 1b6	The retrieval occurs when the referenced object is accessed	The HTTP Gateway inherently maintains a connection cache because it maintains simultaneous connections to multiple W3Object Servers, as shown in Fig. 5					

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
41	See '554 7.	The reference expressly teaches updating databases through user queries: "If the application involves large datasets or updating databases to be maintained at the server, it is not possible to store all the data in HTML forms." Sec. 4.2			The reference teaches a computer-implemented method	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the “server.” The routing is shown by arrows between "HTTP Protocol" and "Encoding"	The request must be received to be processed	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	The request is intercepted and not handled by HTTP (the web server), but is sent to Server (the page server) for processing.	The Web server is shown in Fig. 2 as HTTP. The page server is illustrated in Fig. 2 on page 3 as the “server.” The routing is shown by arrows between "HTTP Protocol" and "Encoding"
42					The reference teaches a computer-implemented method	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5	The page server receives the request in Fig. 5	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	The "HTTP daemon is able to distinguish CGI URLs from standard resources. Upon receiving a request containing a CGI URL, rather than dealing with the request itself the daemon invokes a CGI script identified in the URL." Sec. 2.2. The httpd daemon is the interceptor	The Web server is that portion of the "HTTP Gateway" in Fig. 5 which receives the "HTTP Request." The page server is the W3Object Server of Fig. 5. The routing is shown in Fig. 5

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
41	Because HTTP (the web server) is released per '554 1b3, it is free to process other requests	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as “databases” and “computational services.”	The reference teaches a computer-implemented method			The server gateway must identify which of the “databases” and “computational services it will access	See '554 1d2	
42	Concurrency is inherent in the architecture of Fig. 5. When a request is passed off to a page server, the web server can continue to accept incoming requests	Fig. 5 shows the W3Object Server responding to the request	The objects accessed by the W3Object Server are the data sources	The reference teaches a computer-implemented method	The Web server sends the request to the HTTP daemon (dispatcher)	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	See '554 1b6	The retrieval occurs when the referenced object is accessed	The HTTP Gateway inherently maintains a connection cache because it maintains simultaneous connections to multiple W3Object Servers, as shown in Fig. 5

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
41			See '554 7	See '554 8.	The purpose of HTML in the templates is to derive a format of the data	See '554 7.	The reference expressly teaches updating databases through user queries: "If the application involves large datasets or updating databases to be maintained at the server, it is not possible to store all the data in HTML forms." Sec. 4.2			The reference teaches a computer-implemented method
42										The reference teaches a computer-implemented method

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
41				The request must be received to be processed	Communication of the request from HTTP (web server) to Server (page server) is shown as HTTP (GET, POST) in Fig. 2. HTTP provides acknowledgement that the request has been received, which constitutes TCP releasing	Because HTTP (the web server) is released per '554 1b3, it is free to process other requests	Fig. 1 shows "client-server transactions responding by generating dynamic documents"	Data is retrieved dynamically from data sources shown in Fig. 1 as “databases” and “computational services.”
42	The Web server sends the request to the HTTP daemon (dispatcher)	Dynamic information is required in order for the HTTP Gateway to dispatch the request to the appropriate W3Object Server. "In this environment Web resources are frequently created, moved, and destroyed in an ad-hoc manner, suiting the owner of the information or publishing site." Sec. 21.	The request is passed by the HTTP Gateway to that W3Object Server (Fig. 5) which has the required information. The choice is necessarily informed because different servers access different objects: "One method of interfacing with multiple servers is to make use of an HTTP Gateway, which uses stub objects to forward object invocations through to the appropriate server. The gateway is transparent to clients accessing the objects; incoming requests are simply forwarded to the destination object, which parses the request and replies accordingly. This is illustrated in Figure 5, in which server S1 manages a number of different types of object (illustrated by different shapes) and server S2 manages objects of a single type." Sec. 3.2.3. "The HTTP communication end-point of the server may perform name mapping between URL and the internal name for the object and may also map HTTP requests to appropriate method invocations on the object." Sec. 3.2.1, The reference also teaches load balancing in Sec. 2.1.4	The page server receives the request in Fig. 5	The method of communication with the W3Obect Server (page server) is via HTTP, so TCP releasing occurs, according to EpicRealm	Concurrency is inherent in the architecture of Fig. 5. When a request is passed off to a page server, the web server can continue to accept incoming requests	Fig. 5 shows the W3Object Server responding to the request	The objects accessed by the W3Object Server are the data sources

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
43	173	Knudson, "Application Development with Microsoft's Internet Information Server"	Feb. 2, 1996		'554	The Web server is denominated as such in Fig. 2. "IIS merges the database values into the template files using its special instructions, and sends back a properly formatted HTML file to the client." p. 2	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Interception occurs because the web server knows not to process the request: "IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2
44	016	Gaffney, Illustra's Web DataBlade Module	Mar. 1996		'554	With CGI, programmers build applications that generate static HTML and then access a data source to retrieve content to build the dynamic portion of the Application Page. p. 106	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The request must be received to be processed	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to "requests to the database." If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
43	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	Limitation 1d is met because the released web server processes other requests at least when IIS passes off the database query to SQL Server	Described on p. 3 of the reference: “The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server.”	“The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.” p. 3	The identification is in an IDC file. See '554 1b4	See '554 1d2
44	Dispatcher = "Web Driver" of Fig. 2	The Webdriver “connects to the specified database as a specified user.” p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	Concurrent processing is taught in the reference because there are multiple requests to the database pending, which cannot occur unless the Web Server continues to receive requests while the page server is processing.	The Results Page is dynamically generated in response to the request, P. 107 states, “Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.”	The resulting page includes data retrieved by query from a database (Item 4, p. 110)	The dispatcher (Webdriver) makes a choice based on available databases. "Webdriver handles all aspects of database interface and customizes the applications based on information obtained from a configuration file, the CGI environment, URLs, and HTML forms. Webdriver also manages the database connection and retrieval of the Application Page from the database. The Webdriver also has a GUI page builder to create, edit, delete, and execute HTML pages in Web applications." p. 108.	“Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.” p. 107

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
43		The reference expressly teaches logging in: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.: p. 3		HTX files are custom HTML extension templates	"Finally, IIS merges the database values into the template files using its special instructions, and sends back a properly formatted HTML file to the client. In particular, look at the section of the HTX file delimited with the <%begindetail%> and <%enddetail%> tags. This is where rows returned from the database will appear in the HTML document." p. 4	Microsoft IIS is a network-based systems for handling dynamic Web page generation requests	"The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client." p. 3	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2
44	"Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections," p. 109, which is maintenance of a connection cache to data sources.	Par. 3 on MAC000427 teaches that "the driver connects to the specified database as a specified user." Connecting as a "specified user" inherently discloses the step of "logging into" the data source.	"If your application contains many static large objects, you can improve performance by eliminating some database requests and retrieving large objects directly from the disk cache for Webdriver. When you enable caching, Webdriver creates a disk copy of the large object the first time it is retrieved from the database. Subsequent requests for that large object retrieve the large object from the disk cache. Informix Web DataBlade Module User's Guide, Version 3.3, May 1997, p. 9-3.	"With the Web DataBlade module, a Web-enabled database application becomes a collection of HTML pages containing embedded SQL statements scored in the database. A small (but powerful) set of database-oriented markup tags embed the SQL in the HTML. These Web DataBlade tags adhere to all the properties of SGML processing tags, and at the same time make it possible for you to embed any SQL statement within the Application Page. p. 111. The templates are extensions because they are not HTML but are turned into HTML.	The SQL statements are replaced by the results of the queries into the HTML. "A small (but powerful) set of database-oriented markup tags embed the SQL in the HTML." p. 111. WebExplode "parses the extracted pages, executes the embedded queries, and then formats the results" in the database server. (MAC000427). The embedded queries are SQL statements, which when executed retrieve data from the databases. WebExplode inserts the dynamically-retrieved data into the Application page template. (MAC000427, Fig. 2)	Illustra Web Datablade operates over the Internet, hence is networked. "With CGI, programmers build applications that generate static HTML and then access a data source to retrieve content to build the dynamic portion of the Application Page." p. 106	The resulting page includes data retrieved by query from a database (Item 4, p. 110)	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
43	The first computer system is "The Client." described on p. 1	The second computer system includes IIS and the Internet Database Connector	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2	Interception occurs because the web server knows not to process the request: "IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	The request must be received to be processed
44	The means is the client's Web Browser of Fig. 2	The second computer includes the Web Server, Webdriver and Illustra Server of Fig. 2	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.	Dispatcher = "Web Driver" of Fig. 2	The Webdriver "connects to the specified database as a specified user." p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
43	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Described on p. 3 of the reference: “The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server.”	“The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.” p. 3	Microsoft IIS is a network-based systems for handling dynamic Web page generation requests	Interception occurs because the web server knows not to process the request: “IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	Because the system is a programmed system, it must have a medium on which instructions are stored.	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." “The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL.” "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file.” p. 2	The request must be received to be processed
44	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to “requests to the database.” If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	The Results Page is dynamically generated in response to the request, P. 107 states, “Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.”	The resulting page includes data retrieved by query from a database (Item 4, p. 110)	Illustra Web Datablade operates over the Internet, hence is networked. "With CGI, programmers build applications that generate static HTML and then access a data source to retrieve content to build the dynamic portion of the Application Page." p. 106	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.	The Webdriver “connects to the specified database as a specified user.” p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	Because the system is a programmed system, it must have a medium on which instructions are stored.	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
43	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Interception occurs because the web server knows not to process the request: "IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	Limitation 1d is met because the released web server processes other requests at least when IIS passes off the database query to SQL Server	Described on p. 3 of the reference: "The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server."	'335	The Web server is denominated as such in Fig. 2. "IIS merges the database values into the template files using its special instructions, and sends back a properly formatted HTML file to the client." p. 2	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2	The request must be received to be processed
44	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to "requests to the database." If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.	Dispatcher = "Web Driver" of Fig. 2	The Webdriver "connects to the specified database as a specified user." p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	Concurrent processing is taught in the reference because there are multiple requests to the database pending, which cannot occur unless the Web Server continues to receive requests while the page server is processing.	The Results Page is dynamically generated in response to the request, P. 107 states, "Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted."	'335	With CGI, programmers build applications that generate static HTML and then access a data source to retrieve content to build the dynamic portion of the Application Page. p. 106	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
43	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Interception occurs because the web server knows not to process the request: “IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." “The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL.” "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file.” p. 2	Limitation 1d is met because the released web server processes other requests at least when IIS passes off the database query to SQL Server	Described on p. 3 of the reference: “The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server.”	“The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.” p. 3	The Web server is denominated as such in Fig. 2. “IIS merges the database values into the template files using its special instructions, and sends back a properly formatted HTML file to the client.” p. 2	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.
44	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to “requests to the database.” If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	Concurrent processing is taught in the reference because there are multiple requests to the database pending, which cannot occur unless the Web Server continues to receive requests while the page server is processing.	The Results Page is dynamically generated in response to the request, P. 107 states, “Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.”	The resulting page includes data retrieved by query from a database (Item 4, p. 110)	With CGI, programmers build applications that generate static HTML and then access a data source to retrieve content to build the dynamic portion of the Application Page. p. 106	Dispatcher = "Web Driver" of Fig. 2

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
43	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	The identification is in an IDC file. See '554 1b4	See '554 1d2		The reference expressly teaches logging in: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.: p. 3		See '554 7	See '554 8	The template drives a format of the data: "The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server." p. 3
44	The Webdriver "connects to the specified database as a specified user." p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	The dispatcher (Webdriver) makes a choice based on available databases. "Webdriver handles all aspects of database interface and customizes the applications based on information obtained from a configuration file, the CGI environment, URLs, and HTML forms. Webdriver also manages the database connection and retrieval of the Application Page from the database. The Webdriver also has a GUI page builder to create, edit, delete, and execute HTML pages in Web applications." p. 108.	"Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted." p. 107	"Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections," p. 109, which is maintenance of a connection cache to data sources.	Par. 3 on MAC000427 teaches that "the driver connects to the specified database as a specified user." Connecting as a "specified user" inherently discloses the step of "logging into" the data source.	"If your application contains many static large objects, you can improve performance by eliminating some database requests and retrieving large objects directly from the disk cache for Webdriver. When you enable caching, Webdriver creates a disk copy of the large object the first time it is retrieved from the database. Subsequent requests for that large object retrieve the large object from the disk cache. Informix Web DataBlade Module User's Guide, Version 3.3, May 1997, p. 9-3.	"These Web DataBlade tags adhere to all the properties of SGML processing tags, and at the same time make it possible for you to embed any SQL statement within the Application Page." p. 111	The SQL statements of the templates are replaced by the results of the queries	Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted. p. 107 "When WebExplode finds Web DataBlade tags, it issues SQL statements and formats the results based on your formatting instructions." p. 108

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
43	See '554 7	Updating data sources is inherent in SQL	Object handling extension are expressly disclosed: "Regardless of features, inclusion of this product with the Windows OS will be a fairly compelling argument for its use. Java will be supported in the Explorer, as will distributed OLE objects." p. 1	See '335 13	IIS is computer implemented	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Interception occurs because the web server knows not to process the request: "IIS knows that the IDC file needs to be processed by the Internet Server API, or ISAPI. … The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL" p. 2	The Web server is that component of IIS which receives client requests. The page server is the "Database Connector httpodbc.dll." "The process is kicked off with a straightforward HTML command issued by any browser. The command contains the URL of the IDC file to run … This is then sent via the standard HTTP protocol to the server computer included in the URL." "The IDC file included in the URL is then passed as an argument to the Internet Database Connector, or HTTPODBC.DLL … It's the job of HTTPODBC.DLL to read the IDC file and turn its contents into a SQL request or statement and then send it through ODBC to the database referred to in the IDC file." p. 2
44	A small (but powerful) set of database-oriented markup tags embed the SQL in the HTML. These Web DataBlade tags adhere to all the properties of SGML processing tags, and at the same time make it possible for you to embed any SQL statement within the Application Page." p. 111. If these tags are not used, the templates are HTML templates	Illustra's Web DataBlade module is a comprehensive toolset for creating Web-enabled database applications that dynamically retrieve and update Illustra database content. p. 1	Illustra is object-oriented: "The Illustra object-relational DBMS, is the database of choice for managing the sophisticated content utilized by today's multimedia Web-enabled database applications. With the Illustra ORDBMS you are not restricted to the handful of data types found in ordinary database systems. Instead, you can define your own data types, or use additional data types and access methods provided through other Illustra DataBlade modules." p. 106. The access methods are object handling extensions		Web DataBlade is computer implemented	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2	The request must be received to be processed	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to "requests to the database." If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	The request is intercepted at the Web Server because it is not processed by the Web Server but is instead processed by the Illustra Server.	The "web server" of the claim is "Web Server" of Fig. 2. The "page server" is the "Illustra Server." The routing is shown in Fig. 2

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
43	Limitation 1d is met because the released web server processes other requests at least when IIS passes off the database query to SQL Server	Described on p. 3 of the reference: “The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server.”	“The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.” p. 3	IIS is computer implemented	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	The identification is in an IDC file. See '554 1b4	See '554 1d2	
44	Concurrent processing is taught in the reference because there are multiple requests to the database pending, which cannot occur unless the Web Server continues to receive requests while the page server is processing.	The Results Page is dynamically generated in response to the request, P. 107 states, “Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.”	The resulting page includes data retrieved by query from a database (Item 4, p. 110)	Web DataBlade is computer implemented	Dispatcher = "Web Driver" of Fig. 2	The Webdriver “connects to the specified database as a specified user.” p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	The dispatcher (Webdriver) makes a choice based on available databases. "Webdriver handles all aspects of database interface and customizes the applications based on information obtained from a configuration file, the CGI environment, URLs, and HTML forms. Webdriver also manages the database connection and retrieval of the Application Page from the database. The Webdriver also has a GUI page builder to create, edit, delete, and execute HTML pages in Web applications." p. 108.	“Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.” p. 107	“Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections,” p. 109, which is maintenance of a connection cache to data sources.

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
43	The reference expressly teaches logging in: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.: p. 3		See '554 7	See '554 8	The template drives a format of the data: "The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server." p. 3	See '554 7	Updating data sources is inherent in SQL	Object handling extension are expressly disclosed: "Regardless of features, inclusion of this product with the Windows OS will be a fairly compelling argument for its use. Java will be supported in the Explorer, as will distributed OLE objects." p. 1	See '335 13	IIS is computer implemented
44	Par. 3 on MAC000427 teaches that "the driver connects to the specified database as a specified user." Connecting as a "specified user" inherently discloses the step of "logging into" the data source.	"If your application contains many static large objects, you can improve performance by eliminating some database requests and retrieving large objects directly from the disk cache for Webdriver. When you enable caching, Webdriver creates a disk copy of the large object the first time it is retrieved from the database. Subsequent requests for that large object retrieve the large object from the disk cache. Informix Web DataBlade Module User's Guide, Version 3.3, May 1997, p. 9-3.	"These Web DataBlade tags adhere to all the properties of SGML processing tags, and at the same time make it possible for you to embed any SQL statement within the Application Page." p. 111	The SQL statements of the templates are replaced by the results of the queries	Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted. p. 107 "When WebExplode finds Web DataBlade tags, it issues SQL statements and formats the results based on your formatting instructions." p. 108	A small (but powerful) set of database-oriented markup tags embed the SQL in the HTML. These Web DataBlade tags adhere to all the properties of SGML processing tags, and at the same time make it possible for you to embed any SQL statement within the Application Page." p. 111. If these tags are not used, the templates are HTML templates	Illustra's Web DataBlade module is a comprehensive toolset for creating Web-enabled database applications that dynamically retrieve and update Illustra database content. p. 1	Illustra is object-oriented: "The Illustra object-relational DBMS, is the database of choice for managing the sophisticated content utilized by today's multimedia Web-enabled database applications. With the Illustra ORDBMS you are not restricted to the handful of data types found in ordinary database systems. Instead, you can define your own data types, or use additional data types and access methods provided through other Illustra DataBlade modules." p. 106. The access methods are object handling extensions		Web DataBlade is computer implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
43	The dispatcher is that component of IIS which invokes the Internet Database Connector (IDC) and passes it the request.	"Using this approach, any Web browser can submit an HTML request to IIS using HTTP, but the request is to run a special file with an IDC extension, that contains parameters and commands for the server-based ODBC database." p. 2. The IDC file can be modified dynamically	The dispatcher makes an informed selection of which page server/data source combination to use: "The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database" p .3	The request must be received to be processed	TCP releasing (EpicRealm's contention) occurs because HTTP is used to transfer requests to the page server.	Limitation 1d is met because the released web server processes other requests at least when IIS passes off the database query to SQL Server	Described on p. 3 of the reference: “The template file is a specially formatted HTML document, that basically contains place-holders for data coming back from the server.”	“The IDC file is specially formatted, and contains values that specify what ODBC data source to use, what username to use to log on, what SQL statement to send to the database, and the HTML template to use to send processed results back to the client.” p. 3
44	Dispatcher = "Web Driver" of Fig. 2	See 1a5	The Webdriver “connects to the specified database as a specified user.” p. 110. Thus the dispatcher makes an informed selection and performs dispatching within the meaning of the claim. The selection is informed by the database specification and dynamic load (load-balancing): "The Web DataBlade module also provides Webclient, a tool for managing high-load web sites. Webclient, which can be used as an alternative to Webdriver, is a CGI-based application that can be integrated into any proprietary API that supports CGI. Webclient optimizes connection management between the Web server and the Illustra database. Web site administrators can configure a predefined number of database connections, so that no matter how many CGI processes are invoked, the requests to the database are all managed and processed through these already-established connections. This feature means that administrators can customize their system configuration to optimize the underlying CPU architecture and efficiently handle peak loads." p. 109	The request must be received to be processed	The web server continues to run while Illustra server processes the request. The last paragraph on p. 109 refers to “requests to the database.” If the Web Server were not released to process other requests, there could never be more than one such request pending at any time. "The Web DataBlade module works with al! standard HTTP and standard Web servers." p. 112. HTTP implies TCP, which implies TCP releasing	Concurrent processing is taught in the reference because there are multiple requests to the database pending, which cannot occur unless the Web Server continues to receive requests while the page server is processing.	The Results Page is dynamically generated in response to the request, P. 107 states, “Application Pages are stored in the database. When they are retrieved, the SQL statements are executed and the results are dynamically formatted.”	The resulting page includes data retrieved by query from a database (Item 4, p. 110)

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
45		Oracle WebServer User's Guide Release 2.0 Production, Part No. A23646-1 and Oracle WebServer 2.0 Technical Note			'554	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.	"All inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload the processing of the request to the now initialized WEB Application, prospectively freeing up the HTTP	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.
46	604/605	Oracle WebServer User's Guide Release 2.0.2 Production, Part No. A23646-2 and Oracle WebServer 2.0 Technical Note	Mar. 1996		'554	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	The request must be received to be processed	"All inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
45	“The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension.” ORCL 000764.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. “QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: “LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1=”value1” tag2=”value2” --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point.” ORCL000778-779.	See '554 1b6 (configuration file). "[PL/SQL Agent] also needs to have access to all the details required to actually establish a permanent connection: typically a set of environment variables (ORACLE_HOME and ORACLE_SID for UNIX systems) and perhaps a SQL*Net connect descriptor. All of this information is referred to as a DCD (Database Connection Descriptor), and all of the DCDs known t o the PL/SQL Agent are stored in a configuration file named owa.cfg (DCDs where known as Web Agent Services in Webserver 1.0). . . . DCDs are defined in the PL/SQL Agent's configuration file, but they are activated by creating a directory mapping in the Web Request Broker's configuration file.” ORCL000772.	Oracle WebServer User’s Guide Release 2.0.2 Production, at page ORCL000833 discloses a web page that contains data retrieved in response to user’s request. In particular, column headings and data in the table has been retrieved from a database table. “LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1=”value1” tag2=”value2” --> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point.” ORCL000778-779.
46	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2	See '554 1b6 (configuration file)	"PL/SQL cartridges. These execute stored PL/SQL procedures to generate HTML dynamically using Oracle data." p. 1-3. "If desired, you can use LiveHTML to run the PL/SOL Agent under CGI. and thereby incorporate dynamic Oracle data in hardcoded Web pages." p. 1-17

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
45	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	"LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." p. 1-2. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages" p. 1-16. "LiveHTML code is formatted as SGML comments ... The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." p. 4-30. "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." p. A-8	"include. This command specifies that a file is to be included in the generated HTML page at this point. The file can any of the following: - another LiveHTML file like the current one. - a regular HTML file. - an ASCII file." p. 4-30	OWS 2.0 operates with a Web server, hence is networked	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!-- #command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766. Thus the WRB services are "page servers" of the patent. A processor of a computer that runs WRB services is the page server processing means.
46	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	"LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." p. 1-2. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages" p. 1-16. "LiveHTML code is formatted as SGML comments ... The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." p. 4-30. "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." p. A-8	"include. This command specifies that a file is to be included in the generated HTML page at this point. The file can any of the following: - another LiveHTML file like the current one. - a regular HTML file. - an ASCII file." p. 4-30	OWS 2.0 operates with a Web server, hence is networked	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
45	The first computer is a computer that has a processor that runs a web browser as shown on page 1-3. See ORCL000763. "From a web browser's point of view, Oracle WebServer 2.0 looks just like any other web server." Technical Note, p. 2. Therefore, means for generating said request is a processor of a computer running a web browser as show in the figure at ORCL000763.	The second computer system includes the Web listener, the Web Request Broker, as shown on Technical Note p. 2, and the WRB Executable Engines (page servers)	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766. Means for receiving said request is the processor of a computer that runs the Web Listener. See figure on ORCL000763.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766. Also: the executable code that performs the routing at the second computer system is the router.	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.
46	The first computer system includes the user's web browser. "From a web browser's point of view, Oracle WebServer 2.0 looks just like any other web server." Technical Note, p. 2	The second computer system includes the Web listener, the Web Request Broker, as shown on Technical Note p. 2, and the WRB Executable Engines (page servers)	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
45	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload the	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.	OWS 2.0 operates with a Web server, hence is networked	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	OWS 2.0is a programmed system whose instructions are on a machine readable medium	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.
46	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2	OWS 2.0 operates with a Web server, hence is networked	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	OWS 2.0is a programmed system whose instructions are on a machine readable medium	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
45	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	'335	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.
46	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	'335	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
45	"All inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764.
46	"All inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note, p. 3	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2	"Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." p. 1-7. The "database is used for Web pages that are generated at runtime using "live" data." p. 1-2	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
45	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	See '554 1b6 (configuration file). "[PL/SQL Agent] also needs to have access to all the details required to actually establish a permanent connection: typically a set of environment variables (ORACLE_HOME and ORACLE_SID for UNIX systems) and perhaps a SQL*Net connect descriptor. All of this information is referred to as a DCD (Database Connection Descriptor), and all of the DCDs known t o the PL/SQL Agent are stored in a configuration file named owa.cfg (DCDs where known as Web Agent Services in Webserver 1.0). . . . DCDs are defined in the PL/SQL Agent's configuration file, but they are activated by creating a directory mapping in the Web Request Broker's configuration file." ORCL000772.	Oracle WebServer User's Guide Release 2.0.2 Production, at page ORCL000833 discloses a web page that contains data retrieved in response to user's request. In particular, column headings and data in the table has been retrieved from a database table. "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	LiveHTML comprises tag-based text templates. See '554 7	See '554 8.	!@@Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page
46	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	See '554 1b6 (configuration file)	"PL/SQL cartridges. These execute stored PL/SQL procedures to generate HTML dynamically using Oracle data." p. 1-3. "If desired, you can use LiveHTML to run the PL/SOL Agent under CGI. and thereby incorporate dynamic Oracle data in hardcoded Web pages." p. 1-17	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	LiveHTML comprises tag-based text templates. See '554 7	See '554 8.	Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
45	See '554 8.	"You can also use a cursor to update a table's contents. To do this, use a FOR UPDATE clause to lock the rows in the table. See "Using FOR UPDATE" in the PL/SQL User's Guide and Reference for more information. Sometimes, you may need to use cursor variables. which are not associated with a query until runtime. This is a form of dynamic SQL." p. B-19. UPDATE is similar to SELECT in that it takes a predicate and operates on all rows that make the predicate TRUE. For example: UPDATE Customers SET ADDRESS = null WHERE LNAME = ' Subciak ' ORCL000972.	"The Java Interpreter includes classes for native Oracle7 access via PL/SQL and dynamic creation of HTML documents. For customers who wish to employ their own programming language of choice, Oracle WebServer 2.0 offers an easy-to-use API allowing you to build your own server-side plugins ranging from existing legacy applications written COBOL to modern CORBA or COM/OLE based object-oriented systems written in for instance C++ or Smalltalk." ORCL000762-763. COM and OLE are object-handling extensions, as I understand that term to be used in the Patents	See '335 13. OLE is taught expressly	OWS 2.0 is computer-implemented	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload th	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." ORCL000766.
46	See '554 8.	"You can also use a cursor to update a table's contents. To do this, use a FOR UPDATE clause to lock the rows in the table. See "Using FOR UPDATE" in the PL/SQL User's Guide and Reference for more information. Sometimes, you may need to use cursor variables. which are not associated with a query until runtime. This is a form of dynamic SQL." p. B-19	"The Java Interpreter includes classes for native Oracle7 access via PL/SQL and dynamic creation of HTML documents. For customers who wish to employ their own programming language of choice, Oracle WebServer 2.0 offers an easy-to-use API allowing you to build your own server-side plugins ranging from existing legacy applications written COBOL to modern CORBA or COM/OLE based object-oriented systems written in for instance C++ or Smalltalk." ORCL000762-763. COM and OLE are object-handling extensions, as I understand that term to be used in the Patents	See '335 13. OLE is taught expressly	OWS 2.0 is computer-implemented	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed	The request must be received to be processed	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	"The HTTP engine does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." Technical Note, p. 5. This "handing off" without processing is intercepting.	The Web server is Oracle's Web listener. The page server is a "WRB Executable Engine." "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing." p. 1-2. The WRB sends the request to an appropriate page server. Hence the required routing is performed

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
45	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note,	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!-- #command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a	OWS 2.0 is computer-implemented	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764.	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	See '554 1b6 (configuration file). "[PL/SQL Agent] also needs to have access to all the details required to actually establish a permanent connection: typically a set of environment variables (ORACLE_HOME and ORACLE_SID for UNIX systems) and perhaps a SQL*Net connect descriptor. All of this information is referred to as a DCD (Database Connection Descriptor), and all of the DCDs known t o the PL/SQL Agent are stored in a configuration file named owa.cfg (DCDs where known as Web Agent Services in Webserver 1.0). . . . DCDs are defined in the PL/SQL Agent's configuration file, but they are activated by creating a directory mapping in the Web Request Broker's configuration file." ORCL000772.	Oracle WebServer User's Guide Release 2.0.2 Production, at page ORCL000833 discloses a web page that contains data retrieved in response to user's request. In particular, column headings and data in the table has been retrieved from a database table. "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!-- #command tag1="value1" tag2="value2" --> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache
46	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL" Technical Note,	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2	OWS 2.0 is computer-implemented	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	See '554 1b6 (configuration file)	"PL/SQL cartridges. These execute stored PL/SQL procedures to generate HTML dynamically using Oracle data." p. 1-3. "If desired, you can use LiveHTML to run the PL/SOL Agent under CGI. and thereby incorporate dynamic Oracle data in hardcoded Web pages." p. 1-17	"To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests." p. 1-7. This "pool of processes" is a connection cache

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
45	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	LiveHTML comprises tag-based text templates. See '554 7	See '554 8.	!@@Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page	See '554 8.	"You can also use a cursor to update a table's contents. To do this, use a FOR UPDATE clause to lock the rows in the table. See "Using FOR UPDATE" in the PL/SQL User's Guide and Reference for more information. Sometimes, you may need to use cursor variables. which are not associated with a query until runtime. This is a form of dynamic SQL." p. B-19. UPDATE is similar to SELECT in that it takes a predicate and operates on all rows that make the predicate TRUE. For example: UPDATE Customers SET ADDRESS = null WHERE LNAME = ' Subciak ' ORCL000972.	"The Java Interpreter includes classes for native Oracle7 access via PL/SQL and dynamic creation of HTML documents. For customers who wish to employ their own programming language of choice, Oracle WebServer 2.0 offers an easy-to-use API allowing you to build your own server-side plugins ranging from existing legacy applications written COBOL to modern CORBA or COM/OLE based object-oriented systems written in for instance C++ or Smalltalk." ORCL000762-763. COM and OLE are object-handling extensions, as I understand that term to be used in the Patents	See '335 13. OLE is taught expressly	OWS 2.0 is computer-implemented
46	"A URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection." p. 1-14. "At the lowest level, Oracle; internal API allows us t o establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step." Technical Note, p. 11	"Page caching. Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often." p. 1-5	LiveHTML comprises tag-based text templates. See '554 7	See '554 8.	Section 5 (e.g., page 5-5, MAC003147) describes the use of tag-based text templates in PL/SQL procedures to drive the format of the data in creating the generated page. "The Oracle WebServer Developer's Toolkit is a set of procedures that help the customer create programs to generate dynamic documents that are easily presented in HTML." p. 1-2 (MAC003052). HTML determines the format of a web page	See '554 8.	"You can also use a cursor to update a table's contents. To do this, use a FOR UPDATE clause to lock the rows in the table. See "Using FOR UPDATE" in the PL/SQL User's Guide and Reference for more information. Sometimes, you may need to use cursor variables. which are not associated with a query until runtime. This is a form of dynamic SQL." p. B-19	"The Java Interpreter includes classes for native Oracle7 access via PL/SQL and dynamic creation of HTML documents. For customers who wish to employ their own programming language of choice, Oracle WebServer 2.0 offers an easy-to-use API allowing you to build your own server-side plugins ranging from existing legacy applications written COBOL to modern CORBA or COM/OLE based object-oriented systems written in for instance C++ or Smalltalk." ORCL000762-763. COM and OLE are object-handling extensions, as I understand that term to be used in the Patents	See '335 13. OLE is taught expressly	OWS 2.0 is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
45	"The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." ORCL 000764.	"The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. For instance, the Web Request Broker may be configured to dispatch all URL requests beginning with /java to the integrated Java Interpreter. However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues. The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." ORCL000766. "The Dispatcher continually adjusts the load by controlling how many WRBXs are running at a given time, subject to certain parameters. These parameters are set by the Webserver Administrator, who sets the maximum and minimum number of WRBXs running for each WRB Sei-vice. The Dispatcher creates new WRBXs as needed and connects them to the appropriate WRB Ser	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	The request must be received to be processed. "The PL/SQL Agent performs two major tasks which enable a stored procedure to generate a dynamic HTML page. The first is to launch the procedure and pass in any parameters that were part of the HTTP request." ORCL000771.	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention. The Web Request Broker API document has a diagram at page ORCL01606867 that shows that WEB Application (such as PL/SQL Agent) sends REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. HTTP Server must free such resources because otherwise the REQ COMPLETE Response message has no purpose. ORCL01606867. Under the SBL theory, the INIT Response message sent from Web Application (such as PL/SQL Agent) to HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6. "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]. Also see ORCL01816678-79.	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2. Also: "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments t o the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is t o be included in the generated HTML page at this point." ORCL000778-779.
46	The dispatcher is the "WRB Dispatcher." "Whenever the Web Listener receives a URL that calls for the WRB. it passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines (WRBXs).The Dispatcher finds a free WRBX that is configured to run the desired WRB Service and passes execution to it." p. 1-11	See '554 1b6	The WRB Dispatcher makes an informed selection of which page server (WRBX) should process the request based on dynamic information: which WRBX is free and which one is configured to run the desired service. Load balancing is also performed: "WRB Executable Engine (WRBX). One of a pool of processes that the WRB maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process. WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." p. A-13. "The WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services. ... The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." Technical Note, p. 5	The request must be received to be processed	"AII inter-process communication is handled by the transport independent WRB protocol. Webserver 2.0 supports standard IPC mechanisms" Technical Note, p. 5. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's contention	Concurrency is taught expressly: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing: then it returns to the task of listening for more incoming HTTP requests." p. 1-2. "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP	"As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." p. 6-6	"The Oracle WebServer ... draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data." p. 1-2

	A	B	C	D	E	F	G	H	I	J
8		**CLAIMS AND DEPENDENCIES**			'554	1a	1b1	1b2	1b3	1b4
9					'554					
10					'554					
11					'554					
12					'554					
13		**CLAIM EQUIVALENCE**	**-->**		'554					
47	172	"Microsoft SQL Server in the Active Internet"	Mar. 12, 1996		'554	"many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request."	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or "Other Drivers" (page servers) in Fig. 2	The page server receives the request or it would be unable to process it	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a "Broker:Remote Message Ack event," which is analogous to TCP releasing.	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing
48	601	What is Horizon?	1996		'554	"Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server." p. 2	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The request must be received to be processed	Horizon uses HTTP, which uses TCP, hence TCP releasing.	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently

	K	L	M	N	O	P	Q
8	1b5	1b6	1c	1d1	1d2		
9						2	
10							3
11							
12							
13							
47	Windows NT server routes the request to ODBC, the dispatcher	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	Concurrency is taught expressly in the reference: "The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server."	"By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request."	The "data sources" are the databases of '554 1d1	"The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." It is by reference to these .IDC files that the required identification is made	"The IDC, or *pull model,* allows a Web user to initiate dynamic user driven queries from within an HTML document to a SQL Server, which retrieves or updates information in a database."
48	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	The Web Server Application continues to run while the Horizon Application runs	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	The data sources are the database components accessed during the query	In this case you would probably encode the database identifier for the customer in the URL for that link so that it automatically retrieved data on the correct customer. MM030808	The dynamic retrieval is the SQL query. MM030796

	R	S	T	U	V	W	X	Y
8						9a	9b	9c
9								
10								
11	4	5	6	7				
12					8			
13							Subsumed by '554 1d2	Logically '554 1b1
47	"The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server." Maintaining these connections simultaneously requires a connection cache	ODBC only allows database access to properly logged-in users		"The Internet Database Connector uses two types of files to control how the database is accessed and how the output Web page is constructed. These files are Internet Database Connector (.IDC) files and HTML extension (.HTX) files. The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement. An Internet Database Connector file also contains the name and location of the HTML extension file." "The HTML extension file is the template for the actual HTML document that will be returned to the Web browser after the database information and the extension file have been merged."	The reference expressly teaches custom HTML extension templates: "The Internet Database Connector uses two types of files to control how the database is accessed and how the output Web page is constructed. These files are Internet Database Connector (.IDC) files and HTML extension (.HTX) files. The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement. An Internet Database Connector file also contains the name and location of the HTML extension file."	The reference teaches a networked system	The "data sources" are the databases of '554 1d1	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or "Other Drivers" (page servers) in Fig. 2
48		Passwords are required to access the databases. "If you are running under Windows NT and you make a change to an existing ODBC data source (such as changing the file which it points to or the username and password which it uses) you will need to 'Update' its attributes for Horizon using the Data Sources pane of the Horizon Control Panel." p. 5		A custom extension announced this week translates media objects into Web HTML (Hypertext Markup Language) format. MM 030815	"The behavior of the Horizon Insert action is configured by creating hidden input fields in your HTML form definition. When the form is submitted to the Web server, these hidden fields are sent to the server along with the fields containing the user's input. The Web server then passes the form's fields along to WebDB so it can insert the data and determine the response to send back to the user." p. 6	"Horizon is a set of applications which makes it easy for you to incorporate database access into World Wide Web servers running on Windows NT and Windows 95. p. 1. "Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server." p. 2	The data sources are the database components accessed during the query	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.

	Z	AA	AB	AC	AD	AE	AF	AG
8	9d	9e1	9e2	9e3	9e4	9e5	9e6	9e7
9								
10								
11								
12								
13			Web server of '554 1a1	Logically '554 1b1	Logically '554 1b4	'554 1b5	554 1b6	'554 1b2
47	The first computer system is the machine running the web browser of Fig. 1	The second computer system includes	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or “Other Drivers” (page servers) in Fig. 2	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or “Other Drivers” (page servers) in Fig. 2	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing	Windows NT server routes the request to ODBC, the dispatcher	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which “contain the necessary information to connect to the desired ODBC data source and execute the SQL statement.” Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	The page server receives the request or it would be unable to process it
48	Means is the client computer. "Remote Web client sends a request (form submission or other database query) to a Windows-based Web server." p. 2	The second computer system includes boxes 2-5 in the drawing on p. 2	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	The request must be received to be processed

	AH	AI	AJ	AK	AL	AM	AN	AO	AP
8	9e8	9e9	9e10				11a	11b1	11b2
9				10a	10b	10c			
10									
11									
12									
13	'554 1b3	Subsumed by '554 1d1	Subsumed by '554 1d2	9a	Subsumed by '554 9e4	Subsumed by '554 9e6		Essentially '554 1b1	'554 1b2
47	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a "Broker:Remote Message Ack event," which is analogous to TCP releasing.	"By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request."	The "data sources" are the databases of '554 1d1	The reference teaches a networked system	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	Because the system is a programmed system, it must have a medium on which instructions are stored	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or "Other Drivers" (page servers) in Fig. 2	The page server receives the request or it would be unable to process it
48	Horizon uses HTTP, which uses TCP, hence TCP releasing.	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	The data sources are the database components accessed during the query	"Horizon is a set of applications which makes it easy for you to incorporate database access into World Wide Web servers running on Windows NT and Windows 95. p. 1. "Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server." p. 2	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	Because the system is a programmed system, it must have a medium on which instructions are stored.	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The request must be received to be processed

	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
8	11b3	11b4	11b5	11b6	11c	11d	'335	1a	1b1	1b2
9							'335			
10							'335			
11							'335			
12							'335			
13	'554 1b3	Essentially '554 1b4	'554 1b5	'554 1b6	'554 1c	'554 1d1	'335	'554 1a	Essentially '554 1b1	'554 1b2
47	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a “Broker:Remote Message Ack event,” which is analogous to TCP releasing.	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing	Windows NT server routes the request to ODBC, the dispatcher	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which “contain the necessary information to connect to the desired ODBC data source and execute the SQL statement.” Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	Concurrency is taught expressly in the reference: "The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server."	“By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request.”	'335	“many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request.”	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or “Other Drivers” (page servers) in Fig. 2	The page server receives the request or it would be unable to process it
48	Horizon uses HTTP, which uses TCP, hence TCP releasing.	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	The Web Server Application continues to run while the Horizon Application runs	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	'335	"Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server." p. 2	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The request must be received to be processed

	BA	BB	BC	BD	BE	BF	BG	BH
8	1b3	1c1	1c2	1d	1e1	1e2		
9							2a	2b
10								
11								
12								
13	'554 1b3	'554 1b4	'554 1b1	'554 1c	'554 1d1	'554 1d2	'335 1a	554 1b5
47	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a “Broker:Remote Message Ack event,” which is analogous to TCP releasing.	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or “Other Drivers” (page servers) in Fig. 2	Concurrency is taught expressly in the reference: "The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server."	“By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request.”	The "data sources" are the databases of '554 1d1	“many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request.”	Windows NT server routes the request to ODBC, the dispatcher
48	Horizon uses HTTP, which uses TCP, hence TCP releasing.	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The Web Server Application continues to run while the Horizon Application runs	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	The data sources are the database components accessed during the query	"Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server." p. 2	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).

	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
8									
9	2c	3	4	5	6	7	8		
10								9	10
11									
12									
13	'554 1b6	'554 2	'554 3	'554 4	'554 5	'554 6			
47	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	"The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." It is by reference to these .IDC files that the required identification is made	"The IDC, or pull model, allows a Web user to initiate dynamic user driven queries from within an HTML document to a SQL Server, which retrieves or updates information in a database."	"The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server." Maintaining these connections simultaneously requires a connection cache	ODBC only allows database access to properly logged-in users		These are the .HTX files of the reference	See '554 8.	This is the purpose of the .HTX files
48	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	In this case you would probably encode the database identifier for the customer in the URL for that link so that it automatically retrieved data on the correct customer. MM030808	The dynamic retrieval is the SQL query. MM030796		Passwords are required to access the databases. "If you are running under Windows NT and you make a change to an existing ODBC data source (such as changing the file which it points to or the username and password which it uses) you will need to 'Update' its attributes for Horizon using the Data Sources pane of the Horizon Control Panel." p. 5		Horizon Query makes use of HTML template files to determine the content of pages sent to the user. p. 1. HTML templates are tag-based.	Horizon Query makes use of HTML template files to determine the content of pages sent to the user. p. 1. Data is dynamically retrieved because it comes from a database query.	"You will be able to append data submitted from HTML Forms to tables in your database as well as allow your users to submit queries with flexible criteria and display the results in attractively formatted tables or any other output format which you wish." p. 1. "Once you have defined a query in your template file you can use the return the results of the query to the user in any format which you desire. This is accomplished by inserting the POUTPUT tag in your template file." MM030809

	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA
8					15a	15b1	15b2	15b3	15c1	15c2
9		12	13							
10	11			14						
11										
12										
13						Analogous to '554 1b1	Analogous to '554 1b2	Analogous to '554 1b3	Analogous to '554 1b4	Analogous to '554 1b1
47	The .HTX files include HTML templates	SQL queries can update their data sources dynamically	See '335 14	"Specifically, ActiveX technologies enable developers to build Web content easily using ActiveX controls (formerly OLE Controls), ActiveX scripting, and ActiveX documents." p. 7	The reference teaches a computer-implemented method	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or "Other Drivers" (page servers) in Fig. 2	The page server receives the request or it would be unable to process it	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a "Broker:Remote Message Ack event," which is analogous to TCP releasing.	The request is intercepted at Windows NT Server, not processed by NT Server, but sent on to a different server for processing	Windows NT Server (the web server) routes the request to SQL Server, MS Access Driver or "Other Drivers" (page servers) in Fig. 2
48	See '335 9	Horizon Update - Accepts the input of an HTML form and automatically updates a record contained in an ODBC accessible table. p. 1			Horizon is computer-implemented	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.	The request must be received to be processed	Horizon uses HTTP, which uses TCP, hence TCP releasing.	Intercepting is recognizing a "form submission or other database query," p. 2, and treating it differently	The Web server is the "Web Server Application" on p. 2. The page server is the "Horizon Application" 3 on p. 2.

	CB	CC	CD	CE	CF	CG	CH	CI	CJ
8	15d	15e1	15e2						
9				16a	16b	16c	17	18	19
10									
11									
12									
13	Analogous to '554 1c	'554 1d1	'554 1d2	15a	Analogous to '554 1b5	Analogous to '554 1b6	'554 2	'554 3	'554 4
47	Concurrency is taught expressly in the reference: "The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server."	"By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request."	The "data sources" are the databases of '554 1d1	The reference teaches a computer-implemented method	Windows NT server routes the request to ODBC, the dispatcher	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	"The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." It is by reference to these .IDC files that the required identification is made	"The IDC, or pull model, allows a Web user to initiate dynamic user driven queries from within an HTML document to a SQL Server, which retrieves or updates information in a database."	"The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server." Maintaining these connections simultaneously requires a connection cache
48	The Web Server Application continues to run while the Horizon Application runs	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	The data sources are the database components accessed during the query	Horizon is computer-implemented	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	In this case you would probably encode the database identifier for the customer in the URL for that link so that it automatically retrieved data on the correct customer. MM030808	The dynamic retrieval is the SQL query. MM030796	

	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
8										29a
9	20	21	22				26	27		
10				23	24	25			28	
11										
12										
13	'554 5	Analogous to '554 6	'335 8	'335 9	'335 10	'335 11	'335 12	'335 13	'335 14	'335 15a
47	ODBC only allows database access to properly logged-in users		These are the .HTX files of the reference	See '554 8.	This is the purpose of the .HTX files	The .HTX files include HTML templates	SQL queries can update their data sources dynamically	See '335 14	"Specifically, ActiveX technologies enable developers to build Web content easily using ActiveX controls (formerly OLE Controls), ActiveX scripting, and ActiveX documents." p. 7	The reference teaches a computer-implemented method
48	Passwords are required to access the databases. "If you are running under Windows NT and you make a change to an existing ODBC data source (such as changing the file which it points to or the username and password which it uses) you will need to 'Update' its attributes for Horizon using the Data Sources pane of the Horizon Control Panel." p. 5		Horizon Query makes use of HTML template files to determine the content of pages sent to the user. p. 1. HTML templates are tag-based.	Horizon Query makes use of HTML template files to determine the content of pages sent to the user. p. 1. Data is dynamically retrieved because it comes from a database query.	"You will be able to append data submitted from HTML Forms to tables in your database as well as allow your users to submit queries with flexible criteria and display the results in attractively formatted tables or any other output format which you wish." p. 1. "Once you have defined a query in your template file you can use the return the results of the query to the user in any format which you desire. This is accomplished by inserting the POUTPUT tag in your template file." MM030809	See '335 9	Horizon Update - Accepts the input of an HTML form and automatically updates a record contained in an ODBC accessible table. p. 1			Horizon is computer-implemented

	CU	CV	CW	CX	CY	CZ	DA	DB
8	29b	29c	29d1	29d2	29d3	29e	29f1	29f2
9								
10								
11								
12								
13	Analogous to '335 16b		Equivalent to '554 1b6	'335 15b2	'335 15b3	'335 15d	'335 15e1	'335 15e2
47	Windows NT server routes the request to ODBC, the dispatcher	This dynamic information is contained in .IDC files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement" and can be modified dynamically	ODBC dispatches the request to the appropriate page server. The choice of page server is informed by Internet Database Connector (.IDC) files, which "contain the necessary information to connect to the desired ODBC data source and execute the SQL statement." Because these files can be modified dynamically, the selection is based on dynamic information maintained about page servers. This is Configuration File Dispatching	The page server receives the request or it would be unable to process it	Internet Database Connector (IDC), which is incorporated with Windows NT server (the web server), sends a dynamic ODBC query to SQL Server. SQL acknowledges messages by generating a "Broker:Remote Message Ack event," which is analogous to TCP releasing.	Concurrency is taught expressly in the reference: "The SQL Server Internet Connector allows an unlimited number of Internet connections to a single SQL Server."	"By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request."	The "data sources" are the databases of '554 1d1
48	Dispatching is spawning an instance of the Horizon Application (a page server) to handle a request via Prometheus (Chapter 4).	"If you are running under Windows NT you will also need to 'Publish' the ODBC Data Source so that it is visible to the Prometheus executable when it processes requests from the web server. This can be accomplished using the Data Sources pane of the Prometheus Administrator (PMTHADM.EXE)." MM030800. The Published data sources available to the Prometheus executable are the "data sources a given page server may access."	"Web server launches an instance of Horizon, passing it the data submitted by the client." p. 2. An "appropriate" page server is the one dynamically designated in the query URL (see Chapter 4).	The request must be received to be processed	Horizon uses HTTP, which uses TCP, hence TCP releasing.	The Web Server Application continues to run while the Horizon Application runs	Horizon dynamically generates an HTML page containing the results of the form submission or query and returns it to the Web server. p. 2	The data sources are the database components accessed during the query