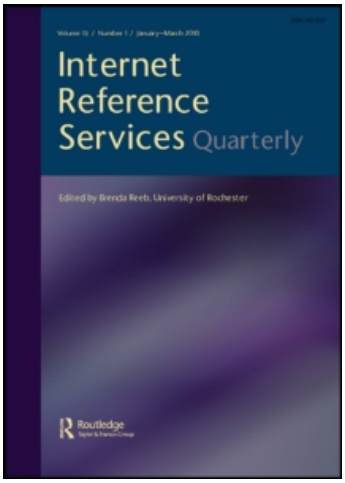



**Internet Reference Services Quarterly**
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t792306879

# HealthLinks: A ColdFusion Web Application

Brian Westra[a]
[a] Hazardous Waste Management Program, Seattle, WA, USA

**To cite this Article** Westra, Brian(2002) 'HealthLinks: A ColdFusion Web Application', Internet Reference Services Quarterly, 7: 1, 63 — 88
**To link to this Article: DOI:** 10.1300/J136v07n01_05
**URL:** http://dx.doi.org/10.1300/J136v07n01_05

# HealthLinks: A ColdFusion Web Application

Brian Westra

**SUMMARY.** Libraries are beginning to use Web applications as they grapple with sites of increasing complexity, and the move of more user services to the Web. This article reviews the basic concepts of a Web application, and outlines some of the features of the HealthLinks Web application and site <http://healthlinks.washington.edu> at the University of Washington Health Science Libraries, and the transition from a Java-based application to ColdFusion. *[Article copies available for a fee from The Haworth Document Delivery Service: 1-800-HAWORTH. E-mail address: <getinfo@haworthpressinc.com> Website: <http://www.HaworthPress.com>* 

**KEYWORDS.** Web application, ColdFusion, database-driven Web site

## *WEB APPLICATION OVERVIEW*

As much of the business world has embraced the Internet, e-commerce innovations and changing user expectations have prompted li-

Brian Westra (brian.westra@metrokc.gov) is Head Librarian, Hazardous Waste Management Program, 130 Nickerson Street, Suite 100, Seattle, WA 98109.

The author would like to acknowledge the following members of the HealthLinks team: Debra Ketchell, Deputy Director, Health Sciences Libraries; Leilani St. Anna, Information Management Librarian; Emily Hull, Head of Information Systems; Adam Garrett, Senior Computer Specialist; Casey Hagan, Student Programmer (ColdFusion database maintenance tools); Cliff Olmsted, Student Systems Administrator (Linux and Apache); and Joanne West, Usability Studies and Webtrends Reports.

[Haworth co-indexing entry note]: "HealthLinks: A ColdFusion Web Application." Westra, Brian. Co-published simultaneously in *Internet Reference Services Quarterly* (The Haworth Information Press, an imprint of The Haworth Press, Inc.) Vol. 7, No. 1/2, 2002, pp. 63-88; and: *Database-Driven Web Sites* (ed: Kristin Antelman) The Haworth Information Press, an imprint of The Haworth Press, Inc., 2002, pp. 63-88. Single or multiple copies of this article are available for a fee from The Haworth Document Delivery Service [1-800-HAWORTH, 9:00 a.m. - 5:00 p.m. (EST). E-mail address: getinfo@haworthpressinc.com].

braries to examine their own online services. In this process, librarians and systems staff are confronted with budgetary and staff constraints, the real or anticipated expectations of users regarding Web-enabled services, and a reluctance to embrace and internalize business and development practices and philosophy. However, libraries are clearly about customer service and any organization interested in improving customer service quality can benefit from examining the successful use of design and technology by excellent service companies.[1] It is therefore worthwhile to note library-oriented articles on portals,[2,3,4] database-driven sites,[5,6] and the confluence of electronic commerce and digital libraries.[7]

In this regard, middleware and Web applications are beginning to see judicious use by some libraries. An exact definition of middleware is difficult to come by, but in a general sense it enables interaction between components (database, e-mail, Web server), and simplifies the programming model for the developer.[8] Web applications are one class of middleware. Web applications can range from "static Web pages, to searchable site/dynamic Web pages, and applications that integrate with operational databases," including customer-driven Web transactions.[9] In a general sense, a Web application is

> a Web system (Web server, network, HTTP, browser) in which user input (navigation and data input) affects the state of the business. This definition attempts to establish that a Web application is a software system with a business state, and that its front end is in large part delivered via a Web system.[10]

These descriptions exemplify the influence that business practices and applications have had upon the terminology. In an excellent review, Fraternelli regards the Web application as "characterized by a direct business-to-customer relationship."[11] On a more concrete level, the technology should attempt to meet some or all of the following requirements:

- handling both structured and non-structured data;
- support exploratory access through navigational interfaces;
- high level of graphical quality;
- customization and possibly dynamic adaptation of content structure, navigation primitives, and presentation styles;
- support of proactive behavior (recommendation and filtering).[12]

Fraternelli points out that these requirements may be in conflict with the following technical and administrative objectives:

- security, scalability, and availability;
- interoperability with legacy systems and data;
- ease of evolution and maintenance.[13]

Web application servers will gain greater acceptance by systems staff as they prove capable of minimizing complexity in dealing with multiple platforms and standards, security, and system-level management functions.[14] Fortunately, a number of products are able to support these needs.

Most models of a Web application are based on a three-layer approach: presentation layer (user-friendly interface), business-logic layer (implements the logic and provides services to the presentation layer), and the system layer, which is responsible for data storage and network requirements.[15] Business logic can also be defined as the piece that "links activities, actors, and resources together within and between companies in buyer-seller relationships."[16] It is relatively easy to extrapolate this definition to the relationship between the library and the patron, e.g., someone using the online circulation systems to renew a book, or setting up a table of contents alerting service for their commonly read journals.

Falk provided some earlier concepts of database-enabled library Web sites or site components that were precursors or have been incorporated into Web applications, such as dynamic page generation, periodical lists, and full-text collections not found in the catalog.[17] These exemplify how Web applications can be used to improve online services to users. Antelman gives an excellent review of the concepts behind library database-driven sites, and the Web management opportunities afforded by Web application servers. Web applications may be a necessary step in building sites of increasing complexity, integrating heterogeneous, distributed resources, while affording some aspects of information management and organization to both library staff and the end user.[18] Application servers such as Macromedia's ColdFusion (tm) provide Web site integration tools, open database connectivity (ODBC) features, and integrated development environments (IDEs)[19,20] that allow developers to more easily construct Web applications.

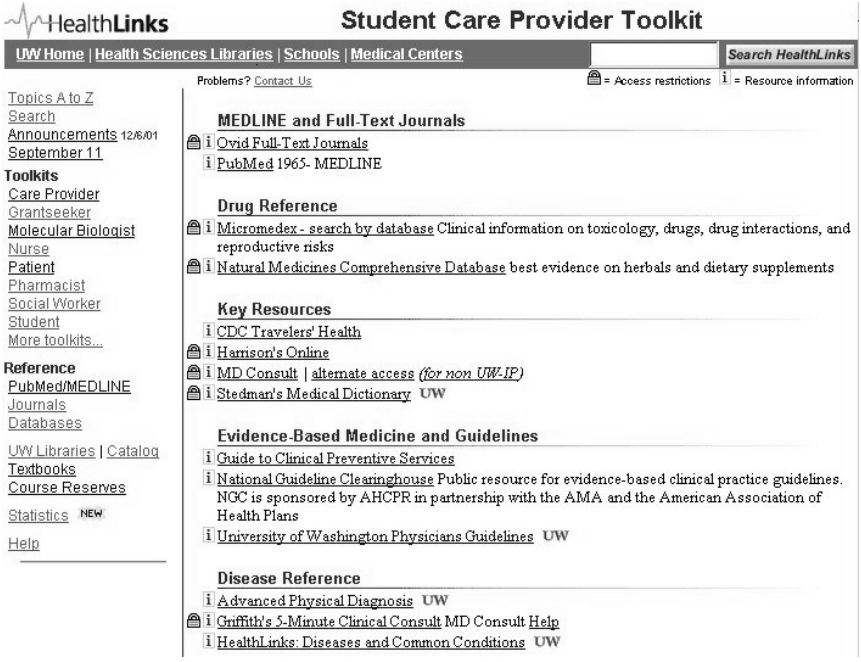
## *HealthLinks OVERVIEW*

The HealthLinks site <http://healthlinks.washington.edu/> is a critical resource for information for faculty, staff and students in the Health Sciences at the University of Washington. The site also serves the various medical centers associated with the university, faculty and students in the five-state WWAMI region (Washington, Wyoming, Alaska, Montana, and Idaho), and the public. Ketchell and Hull have provided a good overview of the site's history, and its utility as a portal for serving the medical clinician.[21,22] The site provides access to textbooks, journals, databases, and other relevant resources for students and faculty, as well as information for the clinician and researcher. Information is presented in the form of role-based "toolkits" (see Figure 1), and topical pages. Each toolkit is developed with and revised in response to user feedback. The site receives significant usage; a three-month average for autumn 2001 indicates the site receives approximately 16,400 hits per day, or 500,000 per month. Work on the site and the underlying database is a collaborative effort, and includes departmental liaisons, serials and systems staff, and others.

Like most library sites, HealthLinks was strictly "hand-maintained" HTML at its inception in 1994, and remained so until 1998. Under funding by an Integrated Advanced Information Management System (IAIMS) grant from the National Libraries of Medicine, a database was developed to maintain several thousand links to commercial and locally developed content. Apache Web server was and continues to be used for static HTML pages and CGI scripts. The move to the use of database content in 1998 was driven by several factors. These included a desire to reduce the number of hand-maintained HTML pages; to facilitate subject experts' access directly to the data and its organization that could be extrapolated to the site, rather than via HTML editing; and to improve search granularity that could not be achieved through a full-site search engine, but might be offered by a database approach.[23]

At the time of the move from flat text to a database-driven site, approximately 150 out of several thousand Web pages of the site were set up to be generated from a database. These 150 pages changed regularly to remain up-to-date and to reflect changing user needs, and were therefore optimal candidates for dynamic or database-generated content. The Java-based Web application wrote out the 150 pages to the Web server, and provided an interface through which library staff could enter and edit records and relationships in the database, which directly affected the organization and content of the generated pages on the site. Distrib-

FIGURE 1. The Student Care Provider Toolkit. This is one of the pages that is generated on a daily basis from the database. The topic is "Student Care Provider Toolkit," the categories are "MEDLINE and Full-Text Journals," "Drug Reference," "Key Resources," "Evidence-Based Medicine and Guidelines," etc. Resources are listed under each of the categories.




uted access to the database maintenance tools was important, since a variety of library staff do data entry and may work from home as well as the office. While this type of Web application is perhaps not as dynamic or synchronous as a shopping cart application for an e-commerce site, it provides a foundation for understanding the site in relation to current terminology. It is worth noting that library staff are also users of the site, as they use the public site and the database maintenance tools to provide instruction and information services and maintain electronic collections and serial subscription information.

The initial Web application was built with JavaServer page utilities and Java servlets to generate the Web pages and provide an interface for the data tools. Page generation was provided by combining information from database queries with HTML tags into text files that were then

written out to the Web server. The database tools allowed staff to edit records and establish relationships in the database, which were used for these page generation queries.

The application utilized IBM Websphere application server[24] and a SQL Server 6.5 database with approximately 5,000 resource records. For the purposes of the HealthLinks database, a "resource" is a unique record, typically pointing to a Web site or specific online source, with information stored in a mixture of Dublin Core metadata and locally produced fields, including URL, title, and associated keywords (see Figure 2).[25] These resources can be grouped together within categories (and subcategories), which are then associated with one or more topics. The basic page of the site that is generated from the database has as its foundation a topic, and from zero to several categories with their respective resources. For instance, in Figure 1, the topic is "Student Care Provider Toolkit," and the categories are "MEDLINE and Full-Text Journals," "Drug Reference," "Key Resources," etc. The resources are the links under each category.

A Java programmer/project administrator and student programmers developed the database and Web application for the site, and it went through iterative changes during the next two years. While the site was successful by many metrics, the Java-based Web application required a programmer to make any changes in the database maintenance compo-



FIGURE 2. Table of Dublin Core metadata and locally produced fields associated with the Resource records.

| **Dublin Core Fields** | **UW Local Fields** |
|---|---|
| Title | Topic |
| Resource Identifier | Category |
| Subject/Keywords | Subcategory |
| Format | Access Restriction |
| Language | UW Resource |
| Author | Class Number |
| Rights Management | Bookplate |
| Description | Help |
| Publisher | Notes |
| Other Contributor | Unique ID |
| Date Published | Record-Created-Date |
| Resource Type | Record-Last-Modified-Date |
| Source | Record-Created-By |
| Relation | |
| Coverage | |

nents or the code that generated the HTML pages. In addition, some of the search capabilities that were an original goal of the transition remained unrealized, though Java could have been used for this feature.

## *CONVERSION PROJECT*

In 2000 it was decided to convert the entire Web application from Java to ColdFusion for the next iteration of the HealthLinks site. This decision was based on several factors, including: the flexibility afforded by the rapid development environment of ColdFusion; relatively low ongoing development cost compared to hiring a Java programmer; more intuitive approach to the code for non-technical staff; utility with both Windows and Linux platforms; and adoption of ColdFusion for Web applications by several other health sciences libraries.

The conversion from Java to ColdFusion focused on three components: (1) static page generation; (2) database maintenance tools; and (3) new dynamic pages and search features as they became possible. Several constraints were placed on this conversion phase. The database schema could not be altered, since the Java-based database maintenance tools would continue to be used while the site generation tools were being replaced by ColdFusion templates. Secondly, while the public site could not be substantially changed in its overall design, changes to improve workflow in the database maintenance tools and new search features for the public site were expected. The new Web application enabled the HealthLinks team to contemplate other components and search features for the site as the project progressed. At recent count, the number of generated pages had risen to 294, and there were 3,667 hand-maintained HTML files in the production site.

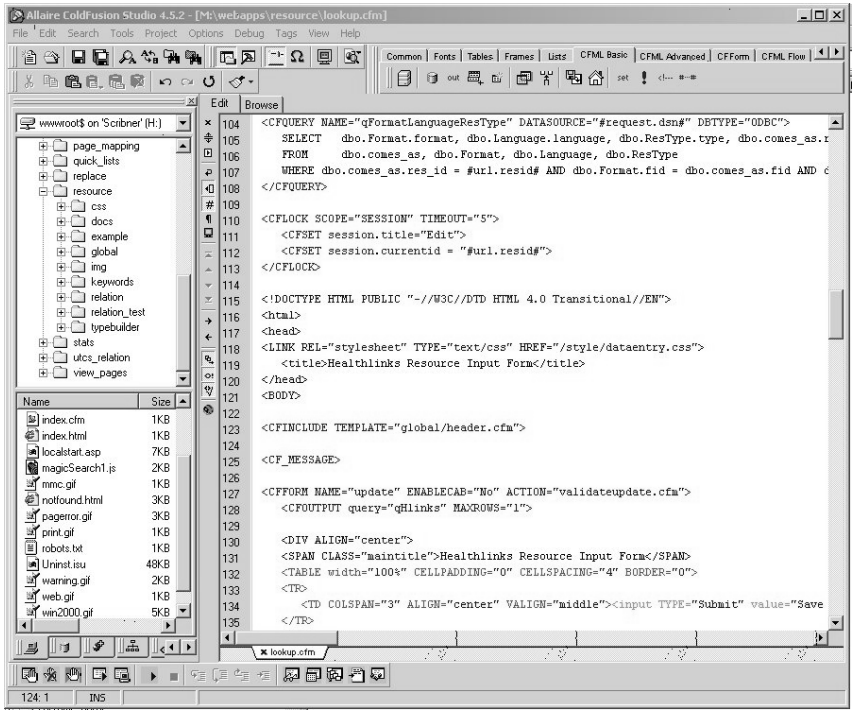
It is good practice to approach a Web application project from the perspective of a development cycle, even if the project itself will be based on rapid development and prototyping. The development cycle emphasizes requirement-gathering, modeling, and prototyping,[26] and is intended to help the development team clarify and meet checkpoints throughout the process, and avoid project "creep." Requirements throughout the project were based on the same site generation capabilities, database access, and workflows that were provided by the Java application. Once the database schema and queries had been established, a "proof of concept" group of templates were developed that could generate the same pages as the existing application. Following this the remaining page generation templates were developed and implemented. The final

steps involved building new data entry tools, and this process was more iterative in nature since workflow improvements could be realized along the way with input by staff and librarians.

### *ColdFusion*

ColdFusion Web application server is produced by Macromedia,[27] and when coupled with the Studio editor, provides a rapid development environment in which coding and scripting, database connectivity and queries, and HTML tags can be quickly combined within a single editor that is easy to personalize and reconfigure (see Figure 3). ColdFusion allows the developer to address issues related to multiple platforms and standards, security, and system management on several levels, in the

FIGURE 3. ColdFusion Studio editor provides an integrated development environment for editing CFML, HTML, and accessing the database tables to quickly build queries.



code that constructs the application, and through configurations in the server administration console.

ColdFusion employs a tag-based approach to programming, where functionality is encompassed within tags rather than needing to be explicitly scripted, and offers some familiarity to those acquainted with HTML, as its namesake ColdFusion Markup Language (CFML) implies. Some understanding of relational databases and structured query language (SQL), and an affinity for basic programming concepts will be useful for those new to application development. Advantages of the use of ColdFusion are the ease and speed of application development due to the tag-based coding features,[28] cross-platform capabilities, a large user base and support forum. Libraries use ColdFusion for various applications.[29-39] Developers are not limited to the present group of CFML tags, since user-defined functions and custom tags can be created for specific purposes, and there is a library of custom tags on the Macromedia site. It is generally easier and faster to create functional applications using ColdFusion than by the use of more complex Web application platforms, such as Active Server Pages (ASP)[40] and PHP, though there are plenty of library applications built with those tools.[41] Ultimately, the choice of an application server is highly dependent on considerations that should include staff skills, training, financial resources, preexisting software and hardware, and a commitment from administration and systems to support the choice.

The basic setup for a ColdFusion application calls for a Web server (UNIX/Linux or Windows OS), with ColdFusion, and an ODBC database on the same machine or another.[42] A typical dynamic page generation occurs in the following way. When the Web server receives a page request that calls for a CFML template or other processing (.cfm or .cfml), it routes the request to the ColdFusion application server. The ColdFusion files ("templates") are written in CFML, and reside on the ColdFusion/Web server. When they are executed they may run queries against the database or do other processing, then combine this information with HTML tagging, and output the resulting file(s) to the client via the Web server.

### *Documentation*

Many Web site developers fail to provide formal documentation, due to the iterative, user-centered approach that is taken with smaller and medium-size sites.[43] This can be true of any development process, and unfortunately, the database and Java-based tools lacked much of the in-

formation that would have eased the conversion. Therefore, another explicitly stated goal of the project was to develop clear documentation. The resulting ColdFusion templates contain numerous CFML comments about specific sections of code, and there are separate documents to provide overviews of the file generation templates, the database, database maintenance tools, and hardware and software configurations.

### *Database*

Because the database and Java code lacked documentation, anecdotal information from the HealthLinks team and examination of the Java code were used to determine how the database schema evolved, and what role the various tables played. Several tables were found to be residuals from older revisions of the database design, and establishing their true role proved to be time consuming. This led to an incremental process for the development of the initial page generation templates, until all relevant tables and relationships were defined.

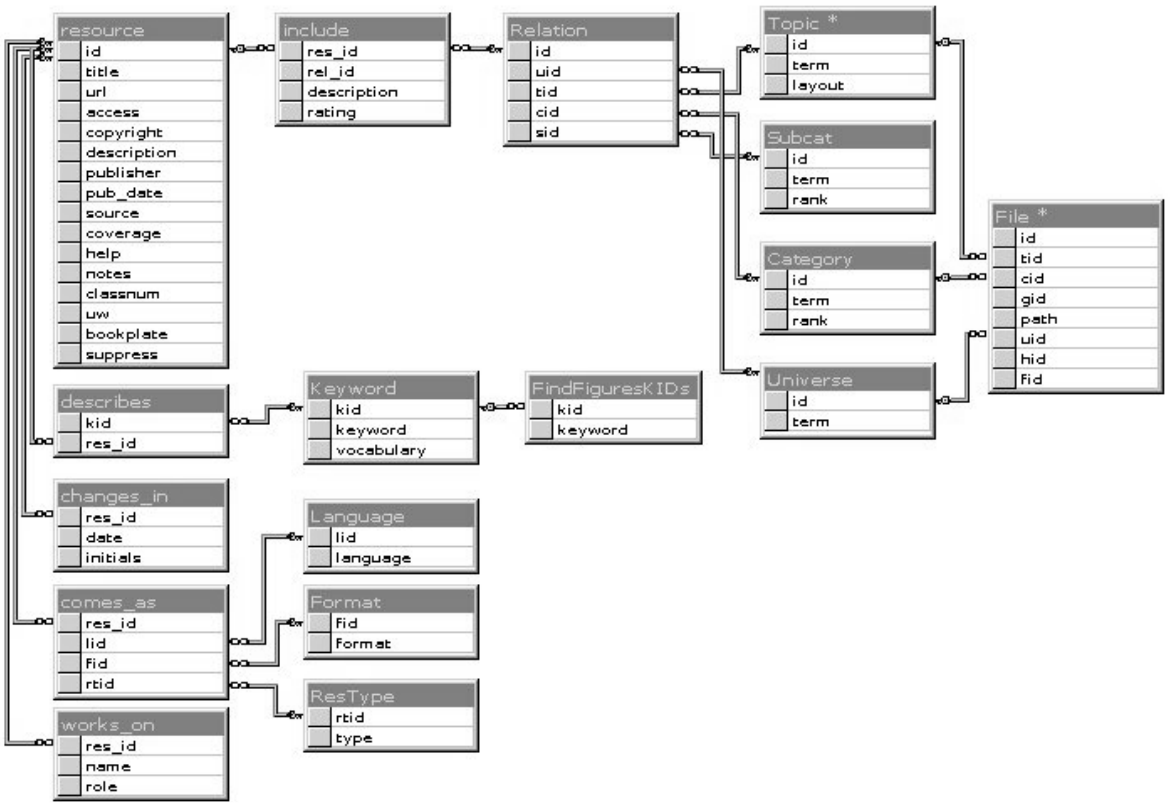
The current database schema is shown in Figure 4. The most basic content component of the HealthLinks database is the resource record. The database diagram shows how resources are related to keywords and other descriptors of the resource. Resources are then related to specific Topic, Topic/Category, or Topic/Category/Subcategory groups, through the Include and Related tables. Further, most pages are generated from the database at the Topic level; that is, most of the pages have a single topic and zero to many categories.

A number of stored procedures are used in the Web application. Stored procedures allow a query to be optimized and stored on the database, rather than in the CFML. This enables the CFML code to be somewhat independent of the database schema, and in databases of larger size and complexity, stored procedures may provide some advantages in overall processing speed for the Web application. As part of the overall project, naming conventions were agreed upon for the stored procedures, and other components within the CFML such as query and variable names.

### *Site Architecture/Hardware*

The ColdFusion Web application was developed in a separate environment from the live site and production database. A copy of the database was exported to a server running Windows NT (later Windows 2000), ColdFusion Web application server and Studio, Apache Web

FIGURE 4. HealthLinks database schema.



server (later Microsoft Internet Information Services 5.0), and Microsoft SQL Server 7.0. In time, a full development architecture was set up to mirror the anticipated production site. In an ideal world, production and development architectures would generally mirror each other, for purposes of working through configuration and site architecture issues, and load testing of templates before moving to the production server. In reality, suitable results for smaller sites with simple database back-ends can often be achieved with single servers for development and production. Some libraries have used file-based databases such as Microsoft Access, but larger and more complex databases and heavy Web loads usually require a client-server database, and segregation of the database server from the ColdFusion/Web server can improve performance.

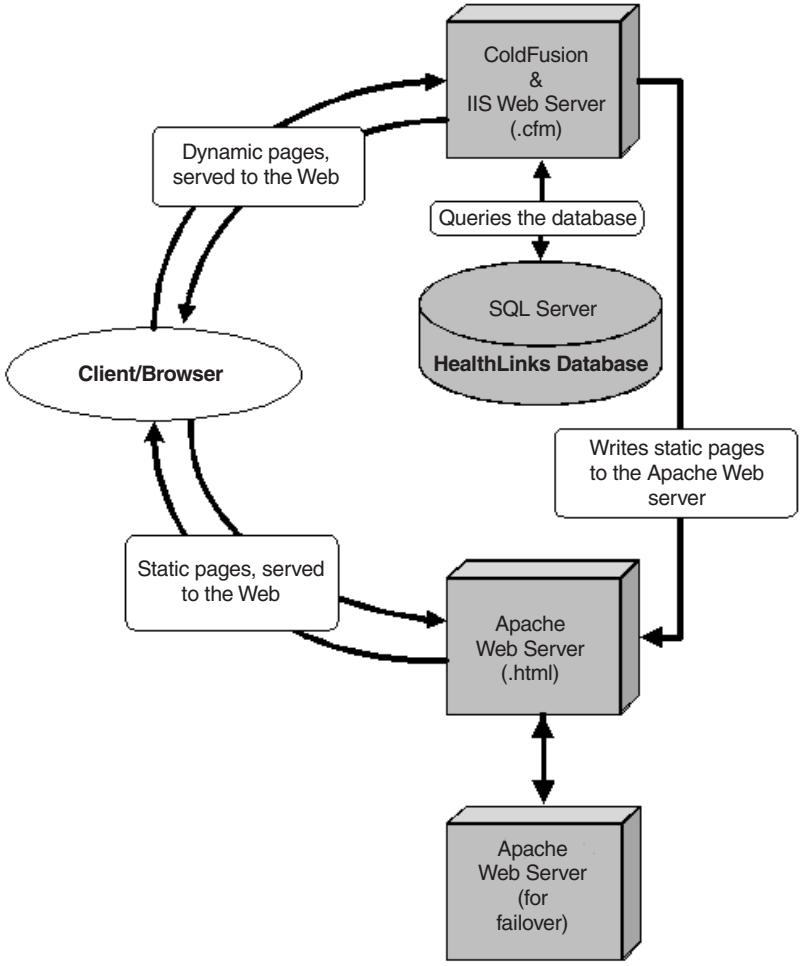
Several older servers which were to be phased out were "recycled" into use for the development site, which was ultimately composed of an Apache Web server running on Linux, and two Windows 2000 servers, one running Internet Information Services (IIS) 5.0 and ColdFusion 4.5, the other devoted to SQL Server 7.0. The production site, which came online in April 2001, has four Dell 2,450 servers, with dual 733 MgHz processors, 1 Gb RAM, and 256 Kb cache. As with most dynamic sites, the greater the RAM, the better the capacity to cache pages and query results, and therefore to handle higher traffic loads. Two servers run Red Hat 7.1 and Apache, with failover between them, for the static HTML and CGI scripts of the HealthLinks site (see Figure 5). Another server runs Windows 2000 and is devoted to ColdFusion/IIS, where the Web application server and Web server run. The database (SQL Server 7.0) resides on the fourth server, also on Windows 2000.

The templates reside on the ColdFusion/IIS production and development servers. Several of the templates call stored procedures on the database, or directly query the database tables, and write out static HTML pages to the Linux/Apache machines. Other templates query the database and populate Verity indexes (called collections) for use with the Verity 97 search engine that comes bundled with ColdFusion. These collections provide full text, Boolean searches, and are run via ColdFusion templates as well.

### *Page Generation, Verity Indexing, and Logs*

Basic queries of the database were tested to determine the correct relationships and optimal SQL statements, and these were later incorporated into the database as stored procedures. By combining the "order

FIGURE 5. HealthLinks production site architecture.






by" statements in the queries and nested output, various configurations of the resulting pages can be achieved.

Four templates have been created to write out static HTML files to the Apache Web server and ColdFusion server for the public HealthLinks site. These templates cover four types of pages, each of which utilizes a different query of the database. The templates generate topic/category pages, e-journal pages for browsing by journal title, topic/category/sub-

category pages (Molecular Biologist toolkit), and statistics resources pages for browsing by topic. These pages may change from day to day, but the information is generally not so fluid as to require "on-the-fly" dynamic generation. In addition, the production and use of these static pages provide information that is up-to-date for the user, while reducing the overall load on the application server and database that dynamic pages would require.

HealthLinks makes use of server side includes (SSIs) to produce the navigational structures on the top, bottom, and side of each page. Apache can be enabled to include SSIs, but a Web server can only employ one type of server side processing on a given page. Since ColdFusion is a type of server side processing, ColdFusion templates can not use the SSI method, but instead use the CFML equivalent, called a <cfinclude> tag, which can point to a file with the .ssi, .cfm, or other extension. This allows the same SSIs to be copied across both Web servers for equivalent content and function in the static and dynamic pages, no matter what the server. In a limited number of cases, certain SSIs for the static and dynamic pages are generated on a daily basis, where the content of the SSI might change from day to day. For instance, the e-journal search box has an alphabetical title browse list, which may change as subscriptions are added or dropped, so this SSI is generated from the database, and written out to both the Apache Web server (for static e-journal pages) and the ColdFusion server (for e-journal title search results pages).

The ColdFusion templates call the stored procedures as specified in their code and use the results to write out HTML files to the appropriate directories on the Apache Web server. The page generation templates are scheduled to run once each day, overwriting the previous day's files. When these scheduled templates run, they also append information to a log file to record template name, query results, and which HTML files were generated.

### *Search Features*

The HealthLinks site is popular, and the use of the Verity search engine that comes bundled with ColdFusion Web Application server has proven to be a scalable approach to searches that meets user needs. In the Java-based Web application, the only search option on the site was a full-site Web search, via a Webinator/Thunderstone search engine. Search results could have been configured to a greater degree, but users found the search results display confusing, and expressed a desire to

have a more targeted search that would yield only the resources from the HealthLinks database, rather than whole Web pages.

Web site searches are now carried out via an Inktomi/Ultraseek search engine, which is run by the University Libraries. ColdFusion templates and the Verity 97 search engine provide the other search options on the site.[44] Any search request that originates on the HealthLinks site, including an Inktomi search, has its search terms recorded by a ColdFusion template before being processed. Hit counts are also recorded for all the Verity searches. This data will enable us to analyze the types of searches (journals vs. other resources), phrase vs. single term searches, and the relationship between search requests and navigational structures and terminology.

The primary search feature is the site-wide resources search, which searches titles and descriptions of resources in the HealthLinks database. The search form is located at the top of all HealthLinks pages, replacing the Webinator search, and therefore has become the default choice. The precise location of the search form and associated text was determined based on a small usability test. There were approximately 18,000 searches per month from September through October 2001; 12,000 were resource searches, 670 were Web site searches, and 5,000 were e-journal title searches.

The e-journal search is a new feature that accompanied the move to ColdFusion. It is accomplished via a Verity collection created by a query of the resource titles related to the e-journal topic. When a search is run against this collection, a list of resource ID numbers for matching e-journal titles is returned to the template. The ColdFusion template then runs a query of the HealthLinks database to return the related information for each resource ID. This information is put into alphabetical order, which is served to the client in an HTML file via the IIS Web server.

The resource search is also conducted against a Verity collection. This collection or index is built from a query that returns the titles and keywords from all "unsuppressed" resource records. Resources can be added to the database, but blocked from inclusion in Verity collections or site generation by means of a checkbox on the Resource data entry form in the database maintenance tools.

Another recently added search is an index of approximately 250 resource records for print and online statistical sources. This search employs a collection of titles, descriptions, and keywords created from a query of only those resources that contain the statistics keyword, which is used to indicate that a resource is primarily statistical in nature. This

keyword is used to create this Verity index, and to generate approximately 130 static pages for browsing by statistical topic.

### *Load Testing*

Load testing is a valuable tool,[45] but many libraries and smaller enterprises do not account for this in their development cycle. A primitive form of load testing was carried out by means of a ColdFusion template that could cycle at a specified time interval. The template ran a basic query of the database that employed relationships between several tables, and therefore provided a good approximation of a typical query. Multiple instances of the template were run, and parameters of the IIS server were logged for analysis. This testing indicated that the application as designed was scalable beyond the anticipated number of hits per minute.

### *Scheduling Template Activity*

Templates for site generation and Verity collection indexing were scheduled to run daily, by means of the ColdFusion Administrator console. These templates and the database maintenance tools are run in a virtual directory with a different port number to isolate them from the search templates for scheduling purposes, and to allow their activity to be logged in a different IIS log than the search template activity. Because the templates are resource-intensive, they were scheduled to run at a time when public use of the site was lowest. However, it was later found that search engine spiders or robots indexing the site would hit all of the information links on a given page within the space of several seconds. Each of those links runs a query and dynamically generates a page. If this occurred at the same time as the CPU-intensive site generation templates were scheduled to run, the Web application server could slow down or lock up. Rescheduling the site generation templates countered this problem. Another possible solution is to generate static pages for all of the information links, rather than pulling them from a dynamic query, or to move more of the information for that part of the site to a database view, which would require less of the database server's resources. Timeout for the templates in the Scheduler is independent of the server-wide timeout setting (which was set to 10 seconds), and some of the scheduled templates take up to 25 seconds to run.

### *Database Maintenance Tools*

*Browser Compatibility*. Because some of the templates in the database maintenance tools make use of JavaScript, it was decided to design this part of the Web application to use Internet Explorer (IE) 5.x, so that template development did not have to follow parallel paths to accommodate differences in how JavaScript is handled by Netscape and IE. This choice is possible since there is a limited population of librarians and staff working with the database tools, and they have ready access to IE on their desktop machines. Since IE was chosen, it was also decided to employ Cascading Style Sheets (CSS) for much of the display features, as CSS enables the developer to quickly and easily alter fonts, colors, and other Web page attributes, and IE does a good job of displaying standards-conformant CSS.

*Pubcookie/UWNetID Authentication*. Distributed secure access was enabled via secure sockets layer (SSL) with a Thawte certificate, and the PubCookie/University of Washington network ID (UWNetID) authentication system. Pubcookie is a centralized authentication system developed by the University of Washington Computing & Communications Department. It is composed of software installed on the server, and the Weblogin server administered by the University.[46] This software is available for Apache and IIS servers. The two components together enable a server administrator to authenticate user access to a particular Web directory by their UWNetID and password, and to set a timeout for this authentication. Upon authentication, a cookie is set. A ColdFusion custom tag was developed which tests the cookie value against a list of authorized UWNetIDs for that particular Web directory. Authorized users are allowed to use the application, while unauthorized users are bounced out to a different directory with an appropriate message. This method is simple, and while it has its limitations, it is adequate to the needs at this time.

Every time a user accesses a file in the database maintenance tools, Pubcookie checks for authentication via the cookie that is saved on the user's machine after the initial login. If the user is authenticated, the next step is that the custom tag is called from within that directory's Application.cfm template. The tag has several attributes that are passed to it in the call, including a list of allowed UWNetIDs, which are compared against the Pubcookie cookie value, and a message to users that are not in the list of authorized users.

*Tools Overview*. The database maintenance tools are a collection of ColdFusion templates aiding staff and librarians in the entry and organi-

zation of data in a SQL server database. The database maintenance tools integrate some of the following features: improvements in resource lookup; data validation rules; context-sensitive user authorization; user timeout; and automatic recording of the user doing record maintenance.

Components of the data entry tools are the ColdFusion templates, SQL Server database, and the Pubcookie software. As with the page generation templates, stored procedures on the database are used throughout this part of the application for increased performance and to isolate some of the database design/schema from the ColdFusion code. Another feature of ColdFusion, transaction blocks, is used for all activities that modify data in the database, whereby a series of queries is either enacted or completely rolled back if it is not completed. This protects data integrity across related tables. Custom tags and server side includes are employed to modularize the code and facilitate code reuse.

The database maintenance tools are not publicly accessible, but several screenshots and descriptions may illustrate some of the more pertinent features. Authorized users can modify resource records, topics, categories, subcategories, relationships between the parts of this hierarchy, and the name and location of the HTML file to be created for a particular topic. The user can also view lists of some of these records, such as the keywords and "orphan" resources that are not currently related to a particular topic but may be important to include for the public site's resource search.

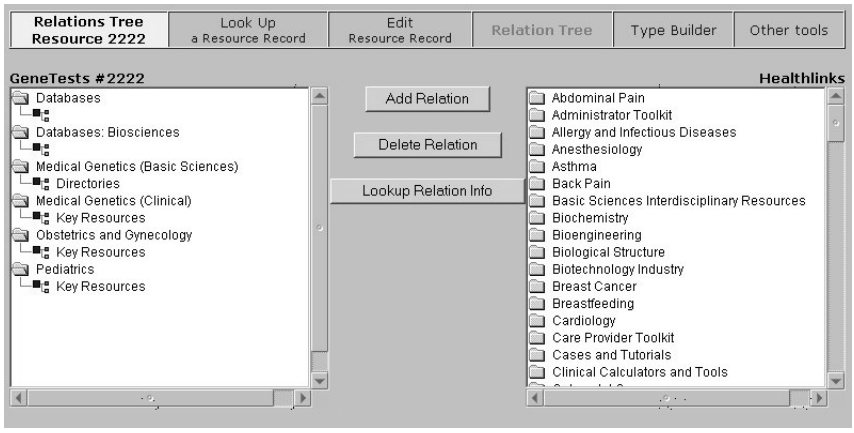
A problem with the Java application became evident as the tables were reviewed. Several tables contained duplicate records, indicating that the business logic had not included sufficient data validation rules for all tables. These were incorporated into the new application, so that duplicates could not be entered. If a term that the staff person is attempting to add to the database already exists, the user is informed that it already is in the database. Otherwise, the term is added to the database, and the user is informed once the update is completed.

*Editing Resources*. The lookup template is the heart of the resource data entry templates, and was revised with input from staff to improve the workflow over the previous search and input forms. All fields are labeled, and required fields have red labels. The only restriction other than a required field is that the URL must be a unique value among resource records in the database. After clicking on the Save button, the fields are parsed and saved into the appropriate tables in the database. The template also checks to see if the resource record in the database has changed since it was delivered for editing.

If the validation process finds that the data has indeed been changed, it redirects the user to a page that displays the original information, along site information submitted by the user. In addition, for those fields where another user has revised the data, that information is shown. If the user wants to go ahead with the update, the template parses the new information again but the check for changed data is not initiated, and anything submitted over-writes the existing record in the database.

*Relations Tree*. One of the more complex display features is the relations tree form. The CFTree tag was used for the relations tree form, because it replicated what was found in the Java Web application, and its Java applet provided the necessary display and data manipulation functions (see Figure 6). In the right window is a CFTree containing all relations in the database that are available to be associated with the resource record being edited. The CFTree applet shows topics (folders), which can be expanded to show existing category components and their subcategories, if they exist. On the left are the current relations associated with the resource, again showing Topic, Category, and Subcategory. When a relationship is added or deleted, the windows update immediately, so the user can see which topics, etc., the resource is related to.

FIGURE 6. Relations Tree form, showing the relation options, and those that have been chosen for resource number 2222, Gene Tests. In this case, the resource has been associated with the Key Resources category, in the Pediatrics topic, among others (see the window on the left).



*Universe, Topic, Category, Subcategory, and Relationship Tools*. Individual universes, topics, categories, and subcategories can be added, deleted, or revised by means of a single group of templates. The universe table and relationships are a carryover from a previous version of the database. HealthLinks is currently the only universe, but this level in the hierarchy may be used at a later time. Code reuse for this group of data tools is made possible by custom tags that call for different functions, based on the information that is passed into them. For instance, working with topics passes the relevant table and query information into a common template that builds a form for either universe, topic, category, or subcategory data entry. Figure 7 shows the topic data entry form, but the same template builds the forms for editing universe, category, and subcategory records.

Whether the user is adding, deleting, or revising a record, a confirmation message is presented as part of the process, and in cases where the record is related to others, the user is given an opportunity to accept or decline the action before it takes place.

*Relationships, Page Mapping, and Viewing Generated Pages*. Several other forms complete the database maintenance tools. After the Topic, Category, and Subcategory terms have been entered into the database, they can be related to each other as a hierarchy (see Figure 8). Once these relationships are established, they will appear in the Relation Tree for the resources, and staff can then relate a selected resource to the particular Topic/Category (and Subcategory) combination.

Pages to be generated are given a specific file name and path in the Page Mapping tool, so they can be written to the Web server (see Figure 9). This information is stored in the File table in the database. Any of the pages to be generated can be previewed within the database tools so that staff can view it before it is generated as part of the production site.

## *CONCLUSIONS*

The conversion to ColdFusion Web application server was successful, and has enabled the HealthLinks team to develop features on the site that had previously been long-term goals. Through their participation, non-developer team members also achieved a better understanding of the database schema, limitations and capabilities of a Web application, and the new opportunities that might be realized in a rapid development environment.

FIGURE 7. The form for editing/revising Topics in the HealthLinks database.

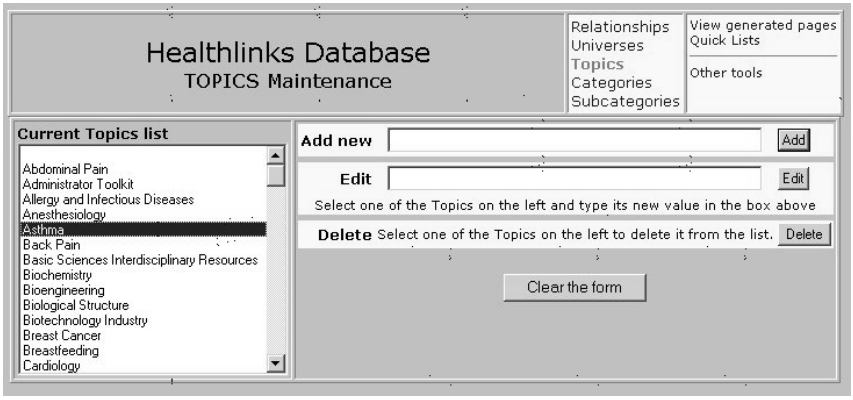
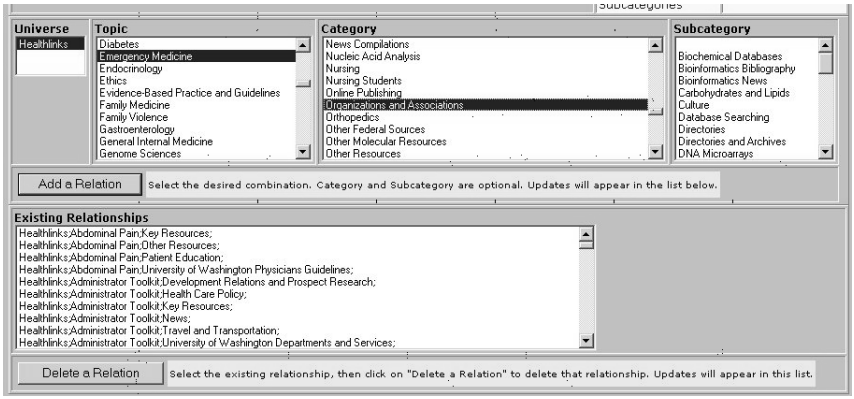
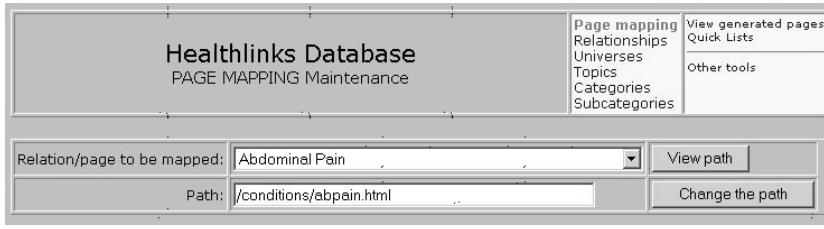



FIGURE 8. Relations tool, showing the relationships that exist (bottom window), and the options that can be selected to create a new hierarchy between universe, topic, category (optional), and subcategory (optional, but requires a category).




Documentation for any technical project is an issue that must be addressed in an ongoing manner. Lack of information can cause critical delays, particularly in environments where the institutional memory is crippled by staff turnover, and this proved to be the case with the transition from the previous HealthLinks Web application. Commenting out code and writing documentation are tasks that most developers would

FIGURE 9. Page mapping tool, to define the path and file name to which a generated page will be written on a daily basis.






rather avoid, as they are time consuming and there is little immediate impact on the development process to show for it. It can also be difficult for an overworked manager to put together a rigorous review of the documentation in preparation for the worst-case scenario. However, project managers should give this component due emphasis throughout the development cycle.

A rapid development environment, such as afforded by ColdFusion, is valuable to the organization if the developer or team can avoid being caught up in a continuous prototype-test-prototype-test cycle. Clear requirements and checkpoints aided in this regard, as did a commitment by project managers to stay on task. An issue for some libraries may be the initial purchase cost for proprietary solutions such as ColdFusion. It is also important to remember the costs of development and long-term maintenance for any given approach, including the time to production, training, and staff and infrastructure requirements.

As has been noted, the resource search in the HealthLinks database became a reality with the new Web application, and it is extremely popular. Approximately 12,000 searches per month in the autumn of 2001 were resource searches. The revised but prominent placement of the search form on all HealthLinks pages, and emphasis on the utility of this new search feature in library instruction, will account to some degree for its popularity. The usability testing associated with the resource and Web searches showed that users are still confused about the differences between these two searches. Only about 670 searches per month were Web site searches, and this option is not as visible as the resource search. It is difficult to find terminology and a search interface that will clearly differentiate these two types of searches, and most users will simply change search syntax or terms, rather than investigate help files or search suggestions.

Other searches recently added to the site include the e-journal title search and statistics resources search. E-journal title searches accounted for another 5,000 searches per month. Logging of search terms and type of search will enable the HealthLinks team to more quickly analyze search strategies and their relationship to terminology, navigation structures, and organizational features. Dynamic page generation also allows staff to include links to URLs with embedded queries for inclusion of "on-the-fly" dynamic content, which previously was not possible.

Custom tags and server side includes are employed to modularize the code to some degree, though more could be done in this area, particularly if a Fusebox coding methodology was followed for the entire application.[47] This approach enables abstraction of the code and code reuse, and may enable future staff to quickly ascertain the functions of each template. However, this approach tends to take longer at the outset, and requires a higher level of programming in the development stage.

Future projects for HealthLinks may include improvements in the indexing terminology, and a refined resource keyword list would lead to improvements in searching.[48] Changes to the database schema may be considered at a later time, as will overall site design modifications. These issues are being investigated by means of other grant-funded projects, which can provide flexible, "test-bed" approaches to the use of the HealthLinks database for applications focused on more specific needs of clinicians.[49,50]



## NOTES

1. Jane Kingman-Brundage, "Technology, Design and Service Quality," *International Journal of Service Industry Management* 2, no. 3 (1991): 47-59.

2. Debra S. Ketchell, "Too Many Channels: Making Sense Out of Portals and Personalization," *Information Technology and Libraries* 19, no. 4 (2000): 175-179.

3. Amos Lakos and Chris Gray, "Personalized Library Portals as an Organizational Culture Change Agent," *Information Technology and Libraries* 19, no. 4 (1999): 169-74.

4. Mick O'Leary, "Grading the Library Portals," *Online* 24, no. 6 (Nov/Dec 2000): 38-44.

5. Kristin Antelman, "Getting Out of the HTML Business: The Database-Driven Web Site Solution," *Information Technology and Libraries* 18, no. 4 (December 1999): 176-181.

6. Bryan H. Davidson, "Database Driven, Dynamic Content Delivery: Providing and Managing Access to Online Resources Using Microsoft Access and Active Server Pages," *OCLC Systems and Services* 17, no. 1 (2001): 34-42.

7. Nabil Adam, Yelena Yesha, Baruch Awerbuch et al., "Strategic Directions in Electronic Commerce and Digital Libraries: Towards a Digital Agora," *ACM Computing Surveys* 28, no. 4 (December 1996): 818-835.

8. Daniel M. Yellin, "Stuck in the Middle: Challenges and Trends in Optimizing Middleware," *ACM SIGPLAN Notices* 36, no. 8 (2001, August): 175-180.

9. Janet Butler, "Internet Applications: No Longer Toys," *Managing System Development* 19, no. 2 (February 1999): 1-9.

10. Jim Conallen, "Modeling Web Application Architectures with UML," *Communications of the ACM* 42 (October 1999): 63-70.

11. Piero Fraternelli, "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey," *ACM Computing Surveys* 31, no. 3 (September 1999): 227-263.

12. Fraternelli, 228.

13. Ibid, 228.

14. Butler, 1.

15. Ezra Ebner, Weiguang Shao, and Wei-Tek Tsai, "The Five-Module Framework for Internet Application Development," *ACM Computing Surveys* 32, no. 1es (March 2000): Article 40, 1-7.

16. Seppo Leminem, "Business Logic in Buyer-Seller Relationships," *Management Decision* 39, no. 8 (2001): 660-665.

17. Howard Falk, "Library Databases on the Web," *The Electronic Library* 14, no. 6: 559-561.

18. Antelman, 176-181.

19. Richard Hoffman, "In the Middle: Enterprise-ready Web App Servers," *Network Computing* 10, no. 11 (May 31, 1999), Available: <http://www.networkcomputing.com/1011/1011r1.html>. Accessed: March 22, 2000.

20. Mark Cyzyk, "ColdFusion Markup Language," *Web Techniques* 5, no. 8 (2000), 3, Available: <http://www.webtechniques.com/archives/2000/08/junk/>. Accessed: December 8, 2001.

21. Debra S. Ketchell, Emily Hull, Leilani St. Anna, Wei-Laung Hu, and Leo Lai, "HealthLinks: Databasing a Web Site," Available: <http://healthlinks.washington.edu/primeanswers/cdl/8icml/>. Accessed: December 17, 2001.

22. Emily Hull, Leilani St. Anna, Kevin Ibrahim, Debra Ketchell, Micah Skilling, Constance Worley, and Steve Rauch, "HealthLinks: Databasing a Web Site," Available: <http://healthlinks.washington.edu/about/history/database/>. Accessed: December 17, 2001.

23. Ibid.

24. IBM, "IBM Software: Web Application Servers: Websphere Application Server: Overview," Available: <http://www-4.ibm.com/software/webservers/appserv/>. Accessed: January 2, 2002.

25. Hull.

26. Ming-te Lu and Wing-lok Yeung, "A Framework for Effective Commercial Web Application Development," *Internet Research: Electronic Networking Applications and Policy* 8, no. 2 (1998): 166-173.

27. Macromedia, "ColdFusion," Available: <http://www.macromedia.com/software/coldfusion/>. Accessed: December 15, 2001.

28. Mark Cyzyk, "ColdFusion Markup Language," *Web Techniques* 5, no. 8 (2000), 3, Available: <http://www.webtechniques.com/archives/2000/08/junk/>. Accessed: December 8, 2001.

29. E.A.B. Draffan and Robbie Corbett, "Implementing a Web-accessible Database," *The Electronic Library* 19, no. 5 (2001): 342-348.

30. Daniel L. Shouse, Nick Crimi, and Janice Steed Lewis, "Managing Journals: One Library's Experience," *Library Hi Tech* 19, no. 2 (2001): 150-154.

31. Ronald C. Jantz, "Providing Access to Unique Information Sources: A Reusable Platform for Publishing Bibliographic Databases on the Web," *Library Hi Tech* 18, no. 1 (2000): 28-36.

32. Arizona Health Sciences Libraries, Ejournals, Available: <http://www.ahsl.arizona.edu/journals/ejrnls/>. Accessed: December 17, 2001; Topics, Available <http://www.ahsl.arizona.edu/guides/topics/>. Accessed: December 17, 2001; Database and Digital Collections Listing, Available <http://www.ahsl.arizona.edu/guides/topics/>. Accessed: December 17, 2001.

33. Ben Forta, Forta.com, "Who's Using ColdFusion?" Available <http://www.forta.com/cf/using/list.cfm?categ_id=4>. Accessed: December 17, 2001.

34. Hofstra University, Available <http://www.hofstra.edu/home/index.html>. Accessed: December 17, 2001.

35. Lamar Soutter Library, University of Massachusetts Medical School, Ejournals, Available: <http://library.umassmed.edu/ejournals.cfm>. Accessed: December 17, 2001.

36. University of Texas Southwestern Medical Center at Dallas Library, Ejournals, Available: <http://www2.utsouthwestern.edu/cfdocs/library/ejournals/ejnls.cfm>. Accessed: December 17, 2001.

37. University of Wisconsin-Madison Health Sciences Libraries, Available: <http://www.hsl.wisc.edu/>. Accessed: December 17, 2001.

38. Virginia Commonwealth University, Course Reserves, Available: <http://www.library.vcu.edu/cfapps/ereserve/index.cfm>. Accessed: December 17, 2001.

39. J.W. Felts, "Now You Can Get There from Here: Creating an Interactive Web Application for Accessing Full-Text Journal Articles from any Location," *Library Collections Acquisitions and Technical Services* 25, no. 3 (Fall 2001): 281-290.

40. Davidson, 34.

41. Felts, op. cit.

42. Allaire Corporation, "ColdFusion 4,5 White Paper," 1999, Available: <http://www.allaire.com/Documents/Objects/WhitePaper/CF45WhitePaper.doc>. Accessed: December 15, 2001.

43. Daniel Cunliffe, "Developing Usable Web Sites–A Review and Model," *Internet Research: Electronic Applications and Policy* 10, no. 4 (2000): 295-307.

44. Sanjay Patel and Charles Linville, "Hot Searches with ColdFusion," *Web Techniques*, Available: <http://www.webtechniques.com/archives/2000/04/patel/>. Accessed: December 15, 2001.

45. Jeff Straathof, "Load Testing Intranet Applications: Finding Hidden Bottlenecks," *Web Techniques* (January 1997): 53-54.

46. University of Washington, Computing and Communications, "Web Authentication Service," Available: <http://www.washington.edu/computing/pubcookie/>. Accessed: December 15, 2001.

47. Fusebox, Inc. Available: <http://www.fusebox.org/>. Accessed: December 15, 2001.

48. Marcia J. Bates, "Indexing and Access for Digital Libraries and the Internet: Human, Database, and Domain Factors," *Journal of the American Society for Information Science* 49, no. 13 (November 1998): 1185-1205.

49. Debra S. Ketchell, Leilani St. Anna, Sherry Dodson, Sara Safranek, and Terry Ann Jankowski, "Knowledge resources: Finding answers to primary care questions," In: *Primary Care Informatics*, Norris et al., ed., Springer-Verlag: in press.

50. Ketchell, "Too Many Channels," 177.

## OTHER RESOURCES

ColdFusion for Libraries Discussion List, Available: <http://faculty.washington.edu/bwestra/cflist.html>. Accessed: 2001, December 17.

Ben Forta, *The ColdFusion 4.0 Web Application Construction Kit*, Indianapolis, Indiana: Que, 1998.