Abstract

Much of the information used in libraries is handled and processed through databases. Often an individual has primary responsibility for data entry of information from many people. If the database has restricted access or is stored on a local machine, then the data entry person may also have the responsibility for retrieving information. Database-to-web technologies can be used to allow individuals throughout the libraries to enter information directly to a central database and retrieve information, removing the additional step performed by a data entry person.

Applications for web interactions with databases can be as diverse as an instruction reporting form, a list of books on approval for faculty review and a subject list of databases.

This paper will provide a brief overview of some of the technologies available to create database-to-web applications, describe three projects which use this approach and mention how those applications can enhance library services and operations. The ability to generate low-maintenance pages and incorporate advanced information technologies enhances services to patrons while freeing staff for other more innovative tasks.

Introduction

A process that has been growing in popularity with libraries over the last several years is using databases to create web pages via database-to-web technologies. This process creates pages “on the fly” or dynamically when requested from the server, as opposed to the traditional static HTML page most of us are accustomed to seeing. Database-to-web technologies have been available in one form or another since the inception of the web and some libraries took advantage of these capabilities years ago. Businesses have been leveraging this technology to create online catalogs and e-commerce features. As software has become available which enables non-computer programmers to develop these type applications, their use in libraries has been increasing.

Database-to-web capabilities have been employed in libraries for several years. This is the process used to create web accessible public catalogs or OPACs. In the last several years these capabilities have become more accessible to non-programmers and thus more widely available for local adaptation by libraries. The article by Antelman provides a good overview of the technologies available for creating dynamic pages. While the article was written several years ago, the options discussed for database-to-web have not changed significantly in that time.
Why would libraries apply this technology locally? The inherent efficiency in re-use of data means databases are a fantastic way to store and present large volumes of data with many common elements. A prime example of this technology in many libraries is the maintenance and display of the collection of electronic indexes. In the context of a libraries’ collection of indexes, a comprehensive database with descriptions and subject designations could be used to generate one list of all indexes and many different subject lists. When a page is rendered, only the necessary pieces are selected for each page display. Different fields in the database can be used for each search if more or less information is to be presented on a specific page, but each page is generated when it is requested, instead of having to be manually created and maintained. Another advantage to this process is URL maintenance for the indexes. Since the information for each resource is stored in a central database, URL changes can be made once in the database and immediately updated on all the pages which include that URL. Many libraries are already using the technology specifically for this application. Libraries using dynamically created pages for maintaining their index information include: University of Southern California, Norris Medical Library, MIT, Cal Poly, University of California San Diego, Ohio State, University of Denver, Regis University and Alfred University.

At Purdue, this technology has been put to work in a data collection application, as well as to present approval book information and to provide electronic access to the contents of our standards collection for the first time.

Not only can lists of material be presented and maintained more efficiently, database-to-web technologies make it possible to create web interfaces for database maintenance, such as adding, editing and deleting records. Creating an administrative web interface saves staff members having to learn the intricacies of the particular database package used to store the information. Other uses for this technology include providing an intuitive public interface for locally developed databases that have not been available to the public in the past, or creating a database to present information that has only been available to the library staff but which is valuable to library users. An example of this at Purdue is automating the approval books review process. The information is already being maintained in a database, for tracking purposes, so presenting a piece of it online is relatively straightforward.

Uses beyond the library specific realm are included in the book Creating Web-accessible Databases, edited by Julie Still. The focus on non-profit organizations highlights how the costs to accomplish this sort of endeavor have become more accessible than compared to a number of years ago. Some of the applications written about are locally developed files for libraries (such as automating local history files), the development of an online used bookstore and the conversion of a paper journal index into a web format. Reading through the chapters provides the reader with a wide view of the capabilities of database-to-web technologies, either in cataloging web pages of electronic books or presenting useful primary research literature in women’s history. Another use of the technology can be seen in indexing web resources about a particular topic as discussed by Draffan and Corbett.

Programming Overview

So how is this accomplished? The technology is available for multiple methods of developing these applications, including CGI (Common Gateway Interface) programming, the Active
Server Page\textsuperscript{13} (ASP) scripting language, PHP\textsuperscript{14} (another scripting language), and middleware packages such as Zope\textsuperscript{15} and ColdFusion\textsuperscript{16}.

In the early days of the web, it was necessary to have a programmer available who could write CGI scripts using either C or Perl. Using CGI programming for database-to-web processes is both labor intensive and resource intensive since each CGI run opens a new process on the server. Options other than writing CGI programs generally include using a middleware package which works between the browser and the web server.

Using middleware reduces some of the programmer knowledge needed to create these applications through a vendor specific application program interface (API). An API is a set of conventions that allow a link within a page to contact a program which is external to the web server. For dynamic pages making use of middleware, the code is embedded into the HTML. When a page is requested, the HTML pieces are handled by the web server and the other parts are passed to the middleware to process and then return the information back to the web server, formatted in HTML. Figure 1 shows the process graphically. The page presented in the browser looks no different to the user viewing the source code than a static page. ASP and PHP code is also embedded in the HTML for a page, but the scripts are executed by the web server.

One middleware option is Macromedia ColdFusion, another is the Open Source product Zope. ColdFusion embeds codes into the HTML using ColdFusion Markup Language (CFML) while Zope uses Document Template Markup Language (DTML). CFML codes are similar to HTML in format and use, but are specific to database functions. DTML is similarly embedded into the HTML for a web page, but its functionality goes beyond just database-to-web and associated
processes. The middleware does much of the interpretation of the CFML or DTML commands, thereby easing the need to know how to program. ColdFusion is packaged with the Verity search engine, creating the ability to index a database and include Boolean searching on a web-accessible database. Similarly, the server part of Zope includes a search engine package. A more thorough look at the core functions in ColdFusion is provided by Beiser.\footnote{Use of any of the options mentioned above require basic knowledge of relational databases and building queries using SQL (Standard Query Language). The SQL commands are written into the dynamic code for the page and describe how the dynamic page is inserting or retrieving information from the database.}

SQL determines the format for requests to the database, also known as queries, to enter information into or retrieve information from a database. Without getting into a complete database and SQL tutorial, information in a database is stored in tables. When you are finding the information in a database that you wish to display, the statement includes what fields you are selecting, what table the information is to be selected from and any parameters that need to be met to refine the results. For example, a database storing information about online indexes may have a table called indexes where the information is stored. Fields in this table could be the database name (name), a description of the database (description) and the URL for access (URL). A couple of sample lines from such a database are shown in Figure 2 for clarity.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. ASCE Civil Engineering Database</td>
<td>Provides access to all ASCE publications published since January 1973. The Civil Engineering Database provides access to over 90,000 bibliographic and abstracted records. This database is available for anyone.</td>
<td><a href="http://www.pubs.asce.org/ceadbarch.html">http://www.pubs.asce.org/ceadbarch.html</a></td>
</tr>
<tr>
<td>5. Catalog Xpress</td>
<td>Facilitates access to vendor catalogs for parts selection. This is an index to the full-text of scanned catalog pages from 15,000 manufacturers.</td>
<td><a href="http://www.ul">http://www.ul</a> sinc.com/cpinteractive/catalog</td>
</tr>
</tbody>
</table>
Statements that put new information into a database table are formatted similarly using an insert command. This command is particularly helpful data collection applications, or when the person maintaining the database is not comfortable in the native database interface. For example, to insert a record for Compendex into our indexes database, the SQL could look like this:

```
INSERT INTO indexes
    (name, description, URL)
VALUES
    ('Compendex', 'A primary engineering database. Covering from 1987-',
    'https://ovid.lib.purdue.edu/directovid?cdexz')
```

It is not necessary for each element of these statements to be on a line of its own, but I find it easier to read the SQL when it is formatted in this fashion.

The programming piece of database-to-web applications does not have to be written by a member of the libraries staff at all, a well supervised student employee with the skills can do the programming. There are pros and cons for this sort of arrangement. Student programming skills are cheap compared to a regular staff member and may actually be more current. However, if the code the student is writing is not well commented (documented), then when that student leaves (graduates or gets a better job on campus), it may be difficult for another person, student or staff member, to decipher the program and make any adjustments or corrections that need to occur with the application. This is a serious concern given the rapid turn-over in student help compared to the life of most library projects and services, and the regularity with which many programmers do not comment the code they write. Even for the programmer it can be difficult to decipher a program later if there are not sufficient notes scattered throughout the code. The advantage to having the programming done by a staff member is the anticipated continuing accessibility of the person who wrote the code and can therefore modify it to make necessary changes or fix problems that have developed. Purdue works with both models depending on the unit in the libraries. For the Engineering Library, the librarian/webmaster is responsible for all the ColdFusion applications that are in use on those pages. The Libraries’ Information Technology Department (ITD) employs student programmers to develop applications on request for libraries on campus.

The choice of software is likely to be made based on what is already available for use in your library or on-campus but can depend on many factors. Many places are using Microsoft’s IIS (Internet Information Server) software which runs on a Windows operating system for their web server. This means choosing ASP, for which no additional software purchase is necessary but more programming time is necessary, Zope, which is Open Source and free, but takes some object-oriented programming knowledge, or ColdFusion, which needs to be paid for but has a low learning curve and allows users to begin working almost immediately. Database-to-web technologies are also available for locations running a Unix or Linux machine generally with Apache server software. Both ColdFusion and Zope can be used with a Unix/Linux set up, so they continue to be options. There is also the possibility of using the free Open Source scripting language, PHP. With Open Source programs there is always the possibility of finding a ready-made and freely available application that serves your needs in one of the many online collections of programs available from locations like www.sourceforge.net.
The server based decision is only one of the considerations when determining what programming language is to be used. The learning curve of people new to the process will influence the choice, as will cost, vendor support and local knowledge. The climate at your location may not be favorable to Open Source products. Open Source products are ones where the source code is available for use or modification by users or developers. Some places may be unwilling to use Open Source software because it does not have the traditional vendor support. However, it does benefit from a community of highly dedicated programmers and users who share their modifications and applications willingly.

Purdue’s choice of ColdFusion was made at a time when ASP and ColdFusion were the leading options for database-to-web programming. Although ASP was known to be more powerful, the desire was to find a product which could be used by more than the “code-warriors” in the technology department. An environmental scan of the University revealed that several departments on campus were using ColdFusion at the time and the company had a strong reputation and a solid product. A significantly discounted academic pricing option kept the cost for ColdFusion from being a barrier to its use at Purdue.

Once the software decisions have been made, or programming options are known, the look and feel of the database and the interface screens can be created. There are several steps to this process and since the information builds in each step, a developer will save time in going through them in order and not jumping directly to the definition of a user interface and coding of the program. Begin with the planning, decide what is needed and wanted in the application and get agreement on the project. Next design the overall system, which includes deciding what information is going to be stored and what interactions are allowed with that data, inserts, updates, deletes, etc. Now it is time to develop the data model. This is a critical step in the process and many future problems can be avoided if proper thought and planning is invested in this step. The goal of this step is to define all the data that needs to be included in the system and then developing the structure to ease maintenance and retrieval. If this step is done thoroughly and well, the data model will handle not only current needs, but should have some flexibility for future needs.

Once the planning, system design and data modeling have been done, the database can be built and data interactions determined. The data modeling should have assisted the designer in determining what data will need to be retrieved, which directly impacts the tables which will store the data and the fields that will be present in each table as well has how the data is entered and retrieved from the database. Now that the necessary interactions and data structure have been determined, the web forms used to enter and present data retrieved from the database, which make up the user interface, can be developed. The final stage is the actual implementation of the project, which includes doing the coding, making sure the forms interact with the database properly and bringing the application out of testing and into service. Depending on the organizational layout in a given company, implementation may also include addressing security issues and data backup questions.

In a recent article, Westman presents the process mentioned above and includes items to be careful of and some discussion of securing the database (which may or may not be part of the concern of the developer). While the article never walks through a specific example, the step-
by-step process outlined would be very useful to anyone beginning this sort of project for the first time, or even as a guide and reminder for someone who has been through the process before. The article is written to cover the entire process for a new project and includes selection of hardware platform, data management system and development tools as part of the implementation phase. As mentioned earlier, many of these decisions may have been made prior to the given project which is in development.

Before committing to any project, it is always wise to consider any drawbacks as well as benefits. Projects using databases need to keep in mind that a large database can slow down the presentation of information being requested from the database. This becomes increasingly obvious with multiple criteria needing to be met during the data selection process. One solution to this is presented in an article by Brown at the University of Denver, where an MS Access database is used to maintain information and generate reports and web pages. Each report is run in Access and then exported as a static web page. Updates can be run daily to ensure the pages are kept up to date, and any data security or response time issues are bypassed by not linking the database directly to the web.

The Purdue experience

Instruction Session Reporting

Now we will look at some examples of how to make the technology work for your library, outside of the maintenance of your collection of databases. At Purdue, statistics on instruction sessions that are offered throughout the year are recorded and tracked. In the past, as each session was completed, the instructor filled out a card which asked for the specifics of the class, including how many students were involved, who the contact person was, and which of our information literacy goals were addressed during the session. These cards were then sent to an individual whose job included entering all of this information into a database for records maintenance. The database was also used to generate reports for each instructor at the end of the year about what they had taught.

Prior to web implementation, the basic pieces were all present. It was already incumbent on each instructor to turn in instruction information when they taught a session or a course. Likewise, the information was already being maintained in a database. Technology can be applied in this situation to free up staff time by removing the data entry tasks from one person. Each instructor has to fill out the information anyway, so why not have the form available online so it can be filled out at the time of the instruction session and have that information go directly to the database, thus removing the need for an extra person to handle and possibly misread the information being entered. There is an added benefit to this method which was not anticipated. The application is written to display a confirmation page which displays in the browser when the form has been completed and the information committed to the database. This shows the instructor exactly what they have added to the file and the page can be printed for individual records maintenance. An example of this confirmation page is shown below in Figure 3.
The online version of the instruction reporting form, shown in Figure 4, was created and made available to the libraries faculty beginning July 1, 2002. Most instructors began using the form immediately and have been quite happy to no longer have to maintain a collection of the paper cards and then remember to fill one out and mail it to someone else for each session. The inevitable changeover period is now occurring, during which some faculty are still using the cards and having that information entered by another person. For many instructors there was a lag time between when the form was announced to the libraries faculty and made available and the first session they taught for the year. This may have contributed to the number of faculty who were still using the paper form at the start of the school year. Due to the length of time the paper form was used prior to the online form being available, some of our libraries faculty may find it easier to fill in the paper form which can be carried to the class with them. After the information from the paper forms was entered into the database, the faculty who had submitted the forms received copies of their confirmation pages and a reminder that the online form was available for use. Aside from easing the data entry, the idea of making the form available electronically was aided by the fact that most of our instruction sessions these days include a computer with
network access, so filling out an electronic form would be as simple as filling in a paper card, and less likely to get misplaced.

A recent discussion of how many departments we are covering in our bibliographic instruction has lead to the inclusion of additional fields on the form to assure consistent data entry. As we began reporting metrics to the university administration as part of our revised strategic plan, it became apparent that we were not consistently tracking classes offered within the same schools and departments. The flexibility of an online form allowed us to add these fields without having to incur the overhead costs that would have been involved in generating new paper reporting cards and ensuring they were distributed to everyone who needed them.
Not only does this process mean a staff member has more time available for other tasks, instead of data entry and report generation, it means instructors are able to generate their own summary reports at need. Instructors can go to a given webpage and select their name from the list of instructors and set the date ranges they would like the report to cover. The report is automatically generated, formatted and presented inside their web browser to be printed.

**Review of Approval Books**

Another project accomplished at Purdue using dynamic pages is the presentation of the approval books for engineering. On a two week rotating cycle there are approximately 40 books available for faculty to stop by the library and review. They may offer comments in the perceived value of the book and request to be notified when it is available to be checked out. As a service to the faculty, a list of these materials was made available on the Engineering Library website, and regular reminders were sent to the faculty when the list was updated.

Every two weeks, after a new shipment arrived, the information about each book was entered into an Access database, which was later used to track the materials through the centralized cataloging process in our decentralized library system, and then exported from Microsoft Access to Excel. Excel was then used to generate a web page, which was then plugged into a static HTML page. This process worked fine and was relatively non-labor intensive until changes made in MS Office 2000 rendered this process unworkable. The code generated by Excel would not import into our web management software. A new method had to be found so large amounts of time were not spent re-keying this information into a web-page, or cutting and pasting from Access.

Since the information was already in a database, it was a relatively simple process to link that database to a web page and generate that page dynamically, eliminating all of the additional steps that used to be necessary to maintain the list of approval books. In a situation like this, with the database already in existence, three things needed to occur. The database needed to be moved to a location with the ColdFusion server could find it, the administrator needed to create a link to the database, and the page to query the database and display the information needed to be written. At this stage, the most difficult part of the display page layout was making it look like the other pages in the Purdue Libraries website. The advantages resulting from this change are fairly obvious. Not only has time been gained on the part of the clerk processing the materials by not having to export into Excel and then generate a web page, there is tremendous time saved for the web site maintainer who no longer has to handle this page to keep it up to date.

**Index of Paper Standards Collection**

A final application, which is in the works at Purdue, will provide access to a local collection for the first time. Our collection of print standards has never had a catalog available to the patrons, or even the library staff. The only way to be sure an item was in our collection was to visit the shelves and check. We began an inventory of the paper standards collection and decided to track basic information for each standard in a database. The process provides a way to determine what is currently in the collection and have a basis for future inventory work. With a database in place, it will be relatively easy to locate all the standards Purdue owns that are published by a given
organization or look up a specific standard number. For the author, the new part of presenting this information on the web will be integrating the search capabilities available with ColdFusion. The Verity search engine is part of the ColdFusion server package. Learning to index the information from a database query and creating a form that accepts search terms, true to other aspects of ColdFusion, seems rather straightforward. While the system in development is not as robust as that developed at Oregon, it will provide an added benefit to our patrons. We’ve begun with a limited database which includes standard number, date, organization abbreviation and the title. The Oregon database includes more fields of information, including designation, revision history, location and description. By virtue of more information present in each record and more fields to be used in refining a search, the application has more access points and more robust search capabilities. In some instance the extra fields would be of no help to another library. At Purdue all the standards we keep in paper are stored in one location, so the location field would be redundant for our purposes.

The advantages of making catalogs of local collections available online can be seen in the articles by Matylonek and Peasely, and Mischo and Schlembach. There is an increase in the use of materials because suddenly it is possible to determine if the library owns a particular item, and in some cases to actually search the database for items with particular keywords.

**Coding tips from experience**

The projects described above are all rather small in scope, and thus performance issues regarding response time or server load have not yet been encountered. Through the process of writing these applications, I learned or remembered several programming practices which will help anyone doing this sort of work, particularly those people who do not write programs on a regular basis.

1) Comment your code! – I mention this in the section on who can write the code and the ability to have continuity between student employees, if they are used. Actually, it doesn’t matter who’s writing the code, it should be commented. No matter how well the programmer knows the application at the end of the development process, several months later when an adjustment needs to be made, the program will have to be relearned just to find the relevant code before any changes can be made. The more guidance that is provided through comments, the faster future changes can be made, and the easier they are to make by someone who didn’t write the program initially.

2) Remember the field types – Within ColdFusion all text fields need to be included in single quotes. When the quotes are forgotten, results will show the first word of a multiple word value or the last group of digits in number sets like phone numbers and social security numbers.

3) Dates – Every database program I have used handles dates a little differently. It’s critical to learn how the database you will be using interprets dates, and how to write your code to collect date information in a fashion that can be properly interpreted and entered into the database.

4) Test the code in chunks – If there are error messages and you are unclear exactly what is causing them, test sections of the page at time. I include additional output commands throughout the program to see what is happening at any given time in the program. When working with form
data, ColdFusion also offers a command that will dump the form contents to a page. This allows
the programmer to ensure the information they are expecting is being collected from the form in
the first place.

5) “Rubber ducking” – This is a process which can be used to help debug a section of code. The
programmer explains the process written in the code, out loud and step-by-step, to an inanimate
object. The name comes from a programmer who originally spoke to a rubber duck on his desk.
It is truly amazing how much this can help find a problem, despite feeling foolish while doing it.

Conclusion

An understanding of relational databases, basic knowledge of SQL and the availability of a
ColdFusion server at Purdue University have allowed the Engineering Library Webmaster to
create functions that ease the process of maintaining multiple pages in an ever-growing website.
By creating methods that use minimal staff time for data entry, and eliminating duplicate data
entry, it has been possible to move more quickly on projects occurring in the library.

The ability to use dynamically generated pages is within the reach of most libraries and the
capabilities of the staff or well-supervised student employees. Benefits can be seen by staff
through time savings and by patrons who will see more consistent and accurate web pages as
well as gain access to information that was not previously available.

References

3. Henning, Nicole. “Improving access to E-journals and databases at the MIT Libraries: building a database-
4. Adalian, Paul T. and Judy Swanson. “Locally developed Web-enabled databases: new roles and opportunities
5. Galvan-Estrada, Laura. “Moving Towards a User-Centered, Database-driven Web Site at the UCSD Libraries.”


12. A brief description of CGI can be found at [http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci213846.00.html](http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci213846.00.html)

13. A brief description of ASP can be found at [http://searchwin2000.techtarget.com/sDefinition/0,,sid1_gci213787.00.html](http://searchwin2000.techtarget.com/sDefinition/0,,sid1_gci213787.00.html)

14. A brief description of PHP can be found at [http://searchenterpriselinux.techtarget.com/sDefinition/0,,sid39_gci334246.00.html](http://searchenterpriselinux.techtarget.com/sDefinition/0,,sid39_gci334246.00.html). Additional information can be found at [www.php.net](http://www.php.net)

15. A brief description of Zope can be found at [http://searchwebservices.techtarget.com/sDefinition/0,,sid_gci213655.00.html](http://searchwebservices.techtarget.com/sDefinition/0,,sid_gci213655.00.html). Additional information can be found at [www.zope.org](http://www.zope.org)

16. A brief description of ColdFusion can be found at [http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci211812.00.html](http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci211812.00.html)


19. A brief description of Open Source software can be found at [http://searchenterpriselinux.techtarget.com/sDefinition/0,290660,sid39_gci212709.00.html](http://searchenterpriselinux.techtarget.com/sDefinition/0,290660,sid39_gci212709.00.html)

20. sourceforge.net and is a repository of Open Source code and applications available on the Internet. Additional Open Source repositories include:


AMY S. VAN EPPS is currently an Assistant Professor of Library Science and Assistant Engineering Librarian at Purdue University, West Lafayette and is the webmaster for the Engineering Library. Prior to coming to Purdue, Ms. Van Epps was the Engineering Librarian for Rensselaer Polytechnic Institute (RPI). While at RPI, Ms. Van Epps earned her Master’s of Engineering in Industrial and Management Engineering. She also holds a Master of Science in Library Science from The Catholic University of America and a Bachelor’s degree in Mechanical Engineering from Lafayette College.