DBMs and Web Delivery

(A more complete version of this text is available at http://www.lib.ncsu.edu/staff/morgan/dbms-and-web-delivery/.)

This text, DBMs and Web Delivery, compares and contrasts three database applications and describes how their content can be made available on the Web.

Filemaker

Filemaker is a cross-platform, desktop, relational database application. The creation of tables is simplistic and the process of creating relationships between them relatively straight forward. If you move or rename the flat tables, then relations will break causing you to recreate the relations.

To say Filemaker is a cross-platform application is a tiny misnomer. Filemaker is supported under various Windows platforms (95, 98, and NT) as well as just about any Macintosh. There is a single difference between Windows and Macintosh versions, and that difference is AppleScript. It allows you to write programs querying Filemaker data and report on it's content. This is great when you want to create something other than simple tab-delimited or rudimentary HTML output from your database. Like many database applications, Filemaker's internal scripting language supports various string, numeric, and date functions for dynamically populating fields or reports.

Besides being extremely easy to use, Filemaker natively supports to essential Internet protocols for effectively providing access to your data via the Web: simple mail transfer protocol (SMTP) and hypertext transfer protocol (HTTP). This means Filemaker can be used as a user agent for sending email and it comes with a built-in Web server.

The implementation of both of these services relies on your ability to write HTML files in Filemaker's specialized mark-up language called Claris Dynamic Markup Language (CDML). CDML is really a server-side include sort of language. Looking a lot like HTML, it allows you to embed commands for adding, editing, deleting, and querying items in your database. As these commands are read by the Filemaker extension enabling HTTP serving they are interpreted and executed against your data. After results are obtained they are formatted into HTML and sent back to the Web browser initiating them in the first place.

CDML's real power, like all other server-side include extensions, is the ability to display dynamically created HTML forms, take the user-supplied input from those forms, and do some processing against the database. For example, you could create a staff directory database and then a CDML file/form listing every staff member's name in a pop-up menu. After selecting an item from the pop-up menu and submitting the form, Filemaker could return more specific information about the staff member including things like telephone number, email address, URL of home page, department, etc. Furthermore, this specific display could be a form itself allowing the authorized user to edit the information as necessary.

Microsoft Access

Microsoft Access is the most popular relational database application here because it is a Microsoft product and it comes bundled with Microsoft's Office suite. Access is more than capable of handling many of your database needs and combined with Microsoft's free Personal Web Server and a knowledge of the Active Server Pages scripting language you can make the content of your Access database(s) available on the Web.

Access has a number of very nice features when compared to Filemaker or MySQL. For one, multiple tables of Access databases are saved in a single file. This makes it more difficult to break your database's relations by simply moving the database file from one hard disk location to another. Similarly, all of Access's queries, forms, reports, and scripts are saved in the same file. The queries are graphically created SQL queries. The forms are database layouts made for each data-entry and querying. Reports are printed outputs of your data. The scripts are VBA (Visual Basic for Applications) modules providing the usual conditional tests, numeric and sting functions, and interfaces with outside applications and the operating system. Finally, Access's database structure reporting mechanisms excel. It is easy to create the standard illustrations describing the databases' structure and database dictionary. Using these functions greatly accelerates technically documenting your database(s) when compared to writing these documents by hand.

Making your Access database content available on the Web is best done through the combination of user-written Active Server Pages (ASP) and an ASP-capable HTTP server. True, Access can export datasets in HTML or rudimentary ASP but it is doubtful you will find the results acceptable because you will end up constantly tweaking the code Access creates and/or the code it creates will be dynamic in only the simplest sense. You are much better writing your own ASP in order to take complete control of the process. Active Server Pages is a server-side include language similar to but more functional than CDML. By inserting special ASP tags in your HTML pages and making sure those pages are passed through an ASP knowledgeable server, the result is dynamically created content.

Microsoft Access is an exceptional database application. It could easily serve as a platform hosting multiple library-related databases. Its primary strength that it is a Microsoft application making it easily integrateable with other Microsoft application. This

also means it is upwards compatible with larger scale applications like Microsoft SQL Server and/or Internet Information Server. The down side of Access is that it is a Windows-only application, and making your data available on the Web requires Microsoft products and a knowledge of ASP.

MySQL

MySQL is a relational database application running on Unix and Windows computers. The Unix version is "open source" and comes with no licensing fees as long as you do not ask for technical support or you do not sell a product requiring MySQL. If you want technical support from the developers, then there is a fee based on a sliding scale. There is also a licensing fee associated with the Windows version of the application. MySQL seems to be the growing favorite in the Internet community.

What MySQL lacks in the user interface department it more than makes up for in functionality and scalabilty. It handles millions of records, supports the majority of SQL functions as well as auto incrementable, variable length, and binary (blob) fields. More importantly, MySQL supports an application programming interface (API) for C, Perl, and PHP.

MySQL sports a minimalistic user interface and a number of programming interfaces. The user interface simply takes its input from a command line. Commands are standard Structured Query Language (SQL) statements: create, use, insert, select, delete, etc. While MySQL supports SQL, it does not support some of the features of its more robust relational database cousins like Sybase or Oracle. Specifically, MySQL does not support stored procedures, rollbacks, or triggers. According to the developers, the most important characteristic of MySQL is speed and these currently unsupported features, if implemented, will diminish its speed.

Command line interfaces are not for everybody and using the APIs supported by MySQL, developers can create applications taking advantage of MySQL and reducing the need to know or understand SQL. The more popular interfaces are C, Perl, and PHP. I have never used the C nor PHP interfaces, but the Perl interface (DBI::DBD for Msql-Mysql) provides all the necessary functionality for creating and maintaining databases.

OBDC

Quite possibly the best solution for putting databases on the Web is not through the database application itself, but through a protocol called Open Database Connectivity (ODBC). ODBC is yet another client server model of computing, and implements an SQL query-results transaction over a network. Essentially a database is set up as an ODBC "data source" allowing it to be queried. The data source acts as the server in a client/server computing model. A client formulates an SQL query and passes it through an ODBC "driver". The driver sends to the query on to the data source, waits for the reply, and finally returns the reply back to the client.

Access, Filemaker, and MySQL can act as servers, and scripting languages like Perl, PHP, or commercial programs like Allaire's ColdFusion or Blue World's Lasso can be used to query the databases. This model is more scalable than many of other techniques described in the previous sections since your ODBC-enabled scripts are oblivious to the databases being queried; the scripts only know about the driver. Therefore, your scripts could stay the same but your databases could move or your data could be migrated to a larger (or smaller) engines.

Conclusion: Databases and libraries

There seems to be no concrete formula for helping you decide what combination of database engines and scripting languages you should implement for making your data available on the Web. Most of the time your decisions will be made for you because you will be limited by particular pieces of hardware and software at your disposal. Even when you do have choices, those choices are hard to make. The largest of database applications such as Oracle or Sybase (often termed "enterprize solutions") will handle any database need but the administrative overhead may be more than you can handle. Desktop applications are easy to use as well as get up and running, but their functionality is limited by the number of records they can easily support as well as the number of simultaneous users.

Libraries are becoming more and more about access (no puns intended) and less and less about storage. People can get their own information and more often than not this information does not necessarily come from a library. In order to provide access to information libraries routinely create lists of things. These lists are organized lists -- they are usually classified with one or more subjects describing the "aboutness" of each item. Using computers, organized lists are best implemented as relational databases.

It behooves librarians to learn and become experts in relational database technology so they (we) can take advantage of the flexibility and particular features of computerized lists. Computerized lists can be updated and copied easily, sorted and re-ordered, limited to subsets of lists, and incorporated into other data formats (ie. word processors, spreadsheets, charts/graphs, etc.) Combined with globally networked computers -- the Internet -- these same lists can be tranfered quickly and flawlessly from one place to another. This process is much a part of librarianship. It is akin to what we have described our role in society to be. Therefore it is appropriate for the us to learn how to use these tools effectively. Otherwise we will be doing our jobs in a manner than is less than professional.

Eric Lease Morgan, NCSU Libraries
http://www.lib.ncsu.edu/staff/morgan/
September 24, 1999