INNOVATION RESEARCH CENTRE

DATA WEBS: AN EVALUATION OF AN INNOVATIVE INFORMATION MANAGEMENT TOOL THAT INTEGRATES DATABASES WITH THE WORLD WIDE WEB

by

R.Bassett, N.P. Archer and W.G. Truscott

Innovation Research Working Group
WORKING PAPER NO. 63

April, 1997

McMASTER
UNIVERSITY

MICHAEL G. DeGROOTE
SCHOOL OF BUSINESS

DATA WEBS: AN EVALUATION OF AN INNOVATIVE INFORMATION MANAGEMENT TOOL THAT INTEGRATES DATABASES WITH THE WORLD WIDE WEB

by

R.Bassett, N.P. Archer and W.G. Truscott

Innovation Research Working Group
WORKING PAPER NO. 63

April, 1997

DataWebs: An Evaluation of an Innovative Information Management Tool That Integrates Databases With the World Wide Web [1]

By

R. Bassett, N.P. Archer, and W.G. Truscott

Michael G. DeGroote School of Business
McMaster University
Hamilton, Ontario
Canada L8S 4M4

[1] This work was supported through a grant from the Innovation Research Centre, Michael G. DeGroote School of Business, McMaster University.

DataWebs: An Evaluation of an Innovative Information Management Tool That Integrates Databases With the World Wide Web

R. Bassett, N.P. Archer, and W.G. Truscott

Michael G. DeGroote School of Business
McMaster University

Abstract

The World Wide Web (Web) has attracted significant amounts of interest from both researchers and business people. Until very recently, nearly all of the pages on the Web were made up of static text and data. Tools and methods have recently been developed which permit the development of applications for use over the Web. The key element to these systems is the ability to access and manipulate data from databases directly from Web interfaces. This paper addresses many of the issues involved with Web/database integration (resulting in systems called datawebs). In particular, traditional client/server models and systems are compared to the Web as an application development environment. It is shown that the datawebs are superior to traditional client/server systems in many regards. Nevertheless, the Web still has some problems associated with it such as concerns with data security and integrity, and this paper discusses a number of these shortcomings. This paper also introduces a sample dataweb which was developed to allow querying and maintenance of an on-line bibliography.

Table of Contents

1. Introduction

The rapid arrival of the World Wide Web (Web) as a business tool and general resource has captured the computing industry by storm. The Web has drawn the attention of many groups of people. The business world has been trying to figure out how to make money from the Web by offering improved or new services over the Web or by selling products on-line. Academic researchers and the system development community have been trying to address pertinent issues such as user interface problems, ever increasing bandwidth demands, and optimal data retrieval schemes. To this end, the technology and the tools necessary to develop Web applications have been developing rapidly over the past couple of years. A major area of application development is how to link existing and new databases to the Web - resulting in systems called datawebs. Many of these dataweb systems are based on existing client/server knowledge, but they are superior to client/server applications in many ways. Comparisons between client/server models and the Web as an application environment will be discussed in this paper. This paper will also address some of the key issues currently being dealt with in these areas of dataweb development. These include the advantages and drawbacks of using the Web, what kind of data and database structures can be accessed over the Web, and some key points to consider to help ensure success in dataweb development. Finally, a sample application will be used to illustrate some of the features of datawebs.

2. The Client/Server Model

2.1 History

The basic concept of client/server technology is simple [20]. The client/server computing model divides the support activities among computers within a companywide or department-level system. The division of labour is apportioned between servers and clients. Servers are typically high-powered computers which often store shared company data. Server functions commonly involve database and administrative application functions such as security and query processing. In contrast, the clients typically contain user-oriented functions such as query entry and data display to the user [20].

On Line Transaction Processing (OLTP) is the automation of high-volume, repetitive business processes with clearly definable transaction steps or procedures. Examples include order processing and invoicing. Evolution of various technologies to support OLTP systems has continued for many years. Beginning in the late 1960's, the traditional computing model began to appear [24]. Prior to this development, batch processing was prevalent, with data commonly entered from computer punch cards. At the heart of this traditional computing model is a powerful host computer which has a number of "dumb" terminals attached to it. These terminals resembled the PC's of today, but they had very little computing power. Another distinction between these dumb terminals and today's PC's is that the dumb terminals were character-based; graphics were not displayed. These early terminals relied on programs that were stored and ran on the host. In the early 1970's, host computers were largely mainframe computers.

Mainframes still play a significant role in systems used today by banks, airlines, telephone companies, etc. The U.S. Department of Defense has about 12,000 mainframes in use, at an average age of 15 years. [24]. The major advantages of mainframe or host-based computing are reliability and security. The major pitfalls are the high operating costs and the length of time required to develop applications.

In the mid 1970's, minicomputers became more popular. These were powerful hosts using the centralized computer model, but had some distinct advantages over mainframes. They were much cheaper than mainframes, allowing organizations the opportunity to purchase more of them and designate specific functions to them. For instance, each department of a large organization could have a designated minicomputer for its computing needs. The big disadvantage with minicomputers was that many of them used proprietary operating systems. For instance, IBM made the AS/400 which used IBM's own OS/400 as an operating system. This made it difficult to transfer application programs from one vendor's computer to another vendor's system.

The increase in the number of minicomputers as well as the introduction of desktop personal computers (PC's) brought on an entirely unforeseen problem. PC's were used not only for desktop computing, but in terminal emulation mode. PC's were recognized as having processing power which was lacking in the dumb terminals of the 1970's. The client/server model was born out of the notion of splitting the processing functions between both the client (the PC) and the server (the centralized system). Client/server systems were developed to overcome some of the drawbacks experienced with mainframe applications. Applications could be developed and deployed more quickly with client/server systems than on mainframes. Graphical displays could be used instead of text displays. Managing growth was easier on a client/server system. There are more standards in place with client/server systems which gave developers more choice with respect to platforms and relational database management systems (RDBMS). Since processing power is split between the client and the server, there is generally less drain on the server but, on the other hand, there is an increase in network traffic which presents another array of problems. Owing to the demands on servers, they can be either mainframes or high-powered workstations.

2.2 Client/Server Models

Today there are primarily four client/server application models being used to develop applications [18]. The four competing paradigms are structured query language (SQL) databases, transaction processing (TP) monitors, groupware, and distributed databases.

The most prevalent of the four client/server models are the SQL databases (Figure 1). Initially, SQL started as a language to manipulate data stored in relational databases. Clients would make data requests from the server by using SQL statements or queries. As the client/server environments became more demanding, it was apparent that simply manipulating data was not enough. It was also necessary to manage the functions that manipulated the data. Stored procedures were designed to group and compile collections of SQL statements. This is just one example of the number of advances with SQL over the years. Although SQL is familiar to a number of programmers and is fairly easy to work with, SQL standards seem to lag behind commercial implementations by about five years [18].

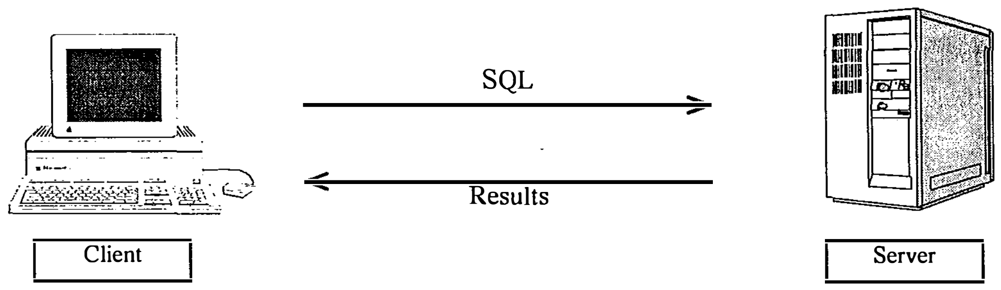

Figure 1. SQL Client/Server Model

Nearly every mission-critical application in the mainframe world comes with a transaction processing (TP) monitor (figure 2). TP monitors are important infrastructure elements in systems that help to ensure that the system delivers reliable and secure operations, and rapid response times [12]. TP monitors provide an extra layer between the server and the client to manage the application processes. TP monitors break complex applications into pieces of code called transactions. An example would be an airline reservation system written for the Internet. A large mainframe reservation system (an example of a legacy system) would already exist to handle the actual reservation, but database applications to capture customer profiles would have to be developed. TP monitors could be used to link the new web database to the existing legacy systems. The existing legacy system would be encapsulated as a set of TP services or functions: get flight schedule, get seat availability, book seat, etc. Thus, customers could update their profile and still access reservation information. Transaction models define when a transaction starts, when it ends, where the processing is handled and what happens when there is a system failure. TP monitors are not dominant in the current client/server applications, but they may become more prevalent as the need to manage the programs themselves becomes increasingly important.

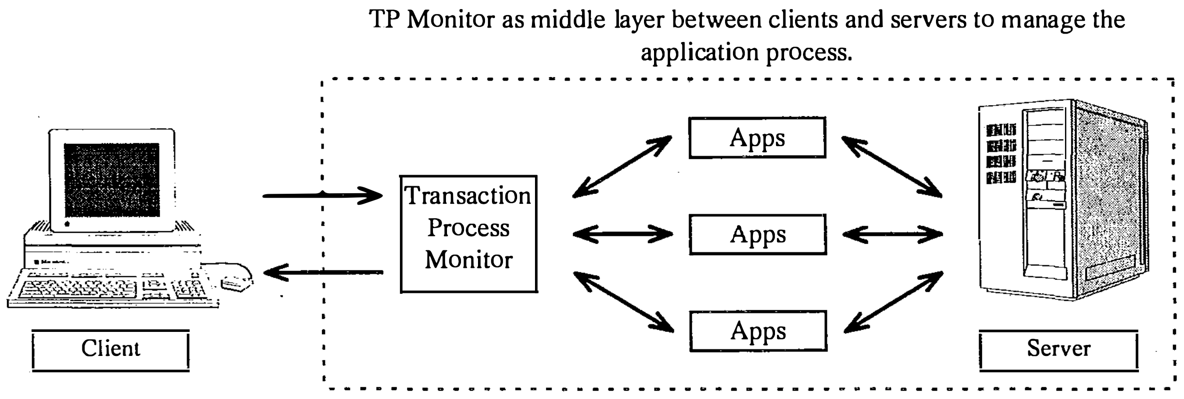
TP Monitor as middle layer between clients and servers to manage the application process.
Client
Transaction Process Monitor
Apps
Apps
Apps
Server

Figure 2. Transaction Process Monitor Client/Server Model

The third client/server model is groupware, which contains the necessary technologies to support collaborative work. These technologies are multimedia document management, work flow, e-mail, conferencing and scheduling. As shown in Figure 3, these technologies can run on the client and/or the server. Groupware helps users share and collect unstructured data such as documents and spreadsheets. The key player in the groupware market is Lotus with its Lotus Notes. One of the major reasons for the success of Notes is its ability to manage document databases in a client/server fashion.

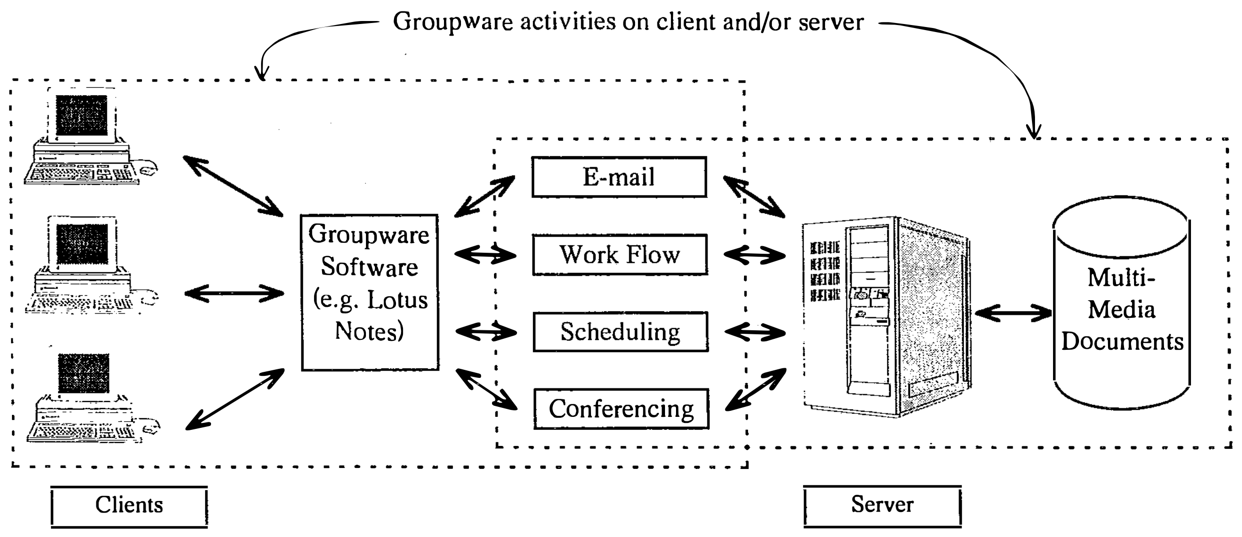
Groupware activities on client and/or server
Groupware Software (e.g. Lotus Notes)
E-mail
Work Flow
Scheduling
Conferencing
Multi-Media Documents
Clients
Server

Figure 3. Groupware Client/Server Model

The last client/server model to consider is distributed-object technology. This approach is more flexible than the other client/server approaches because it encapsulates data and business logic into smaller objects. These objects can be designed to act as agents, they can run on different platforms, and they can manage themselves and the resources they run. The key result is that this approach can break very large programs into more manageable components. These components can be distributed to the client and/or the server. As will be discussed shortly, this is very important for developing Web-based database applications. A significant advantage of this approach over the other client/server models is that standards are already in place. In other words, the standards preceded the widespread implementation of these types of systems, which is quite the opposite of the situation with SQL based systems.

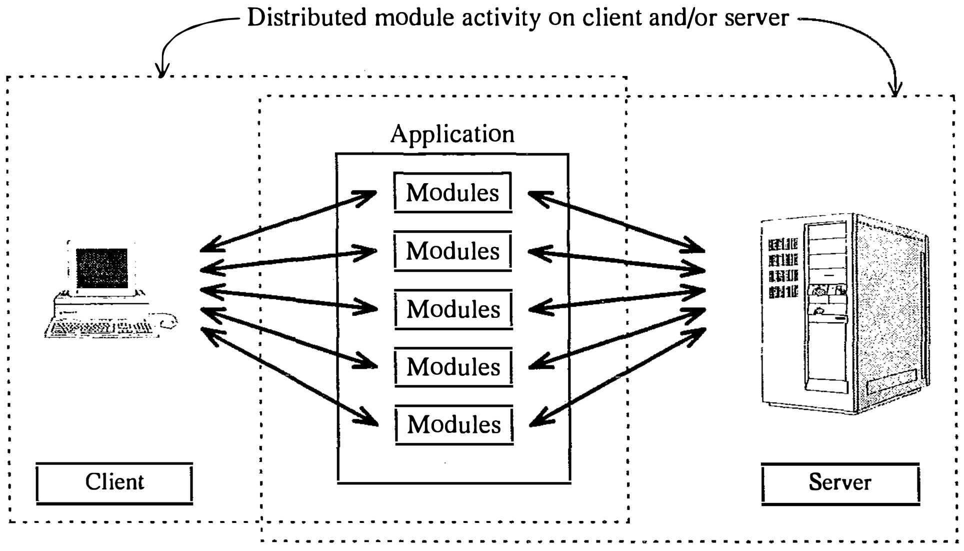

Figure 4. Distributed Objects Client/Server Model

In addition to being classified into the four different models above, client/server systems can be described as being two-tiered or three-tiered. Two-tiered systems involve the presentation and data layers only (Figure 5). The SQL is either hidden beneath the graphical user interface (GUI) or is executed as stored procedures in the DBMS. Most of the PC-based 4GL tools, such as Visual Basic and PowerBuilder, create applications based on a two-tiered architecture [10]. These tools attempt to isolate non-technical users from the complexities of SQL coding. This model may be acceptable with fairly simple systems with relatively small databases being stored and accessed (less than a gigabyte), but for larger systems with complex analytic requirements this model may not be adequate. The reason is that SQL is not a very robust programming language. It is especially weak in doing data comparisons [10]. This is where a three-tiered architecture comes in. This approach strictly enforces a logical separation of the GUI, the business logic, and the data (Figure 6). Generally, this approach allows for more complex analysis because the system is limited neither by SQL nor by the underlying DBMS. In addition, the three-tiered systems are more able to scale upward to larger databases then the two-tiered systems.

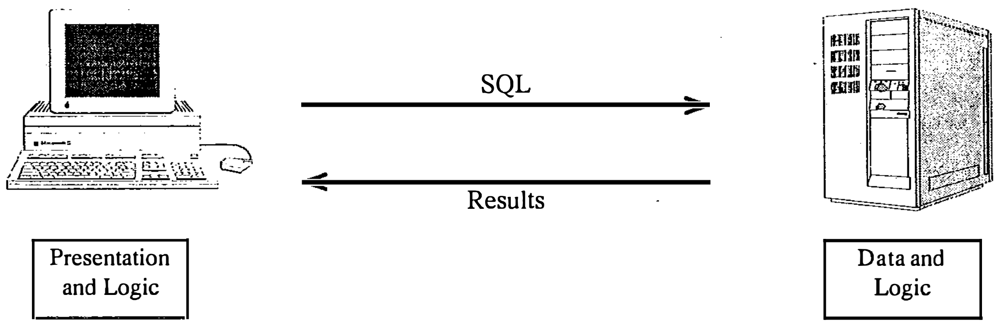

Figure 5. Two-tiered Client/Server System

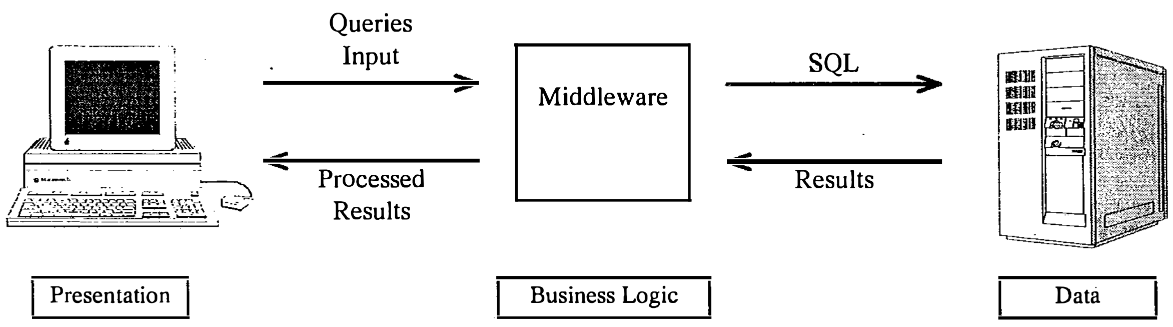

Figure 6. Three-tiered Client/Server Model

2.3 Advantages and Disadvantages of Client/Server Computing

There are a number of compelling forces which have attracted organizations to the client/server model [19]. A major motivation is the desire for end users to obtain fast access to both internal and external databases. Internal databases generally refer to databases maintained within an organizations, whereas an external database would not be. To use a human resource department as an example, a database holding information on current employees would be an internal database. On the other hand, an on-line database of potential employees maintained by an employment agency but which can be accessed through the Internet would be an example of an external database. Well developed systems offer the user transparent access to the various databases necessary for the particular application. Another motivation is increased management control. The introduction of PC's into organizations brought with it a number of management problems such as a lack of control over development, data integrity, and the acquisition of incompatible software and hardware. The client/server model centralized control of data and application development. Other reasons for the attraction towards client/server computing include the sharing of hardware and software resources, faster development, and reduced application development backlog. Of particular interest is that the development backlog may not have been decreased at all, for there may actually be an increase in the number of applications demanded. This has been offset to some extent by the fact that many modern application packages are purchased and not developed locally.

These advantages of client/server over mainframe systems (otherwise known as legacy systems) are significantly offset by a number of drawbacks. Software distribution is a major concern. If a system consists of 1,000+ clients and an upgrade is required to the client side, the task could be very draining on time and resources. Another concern is version control. In this 1,000+ client system, monitoring which versions of different software that exists on the various clients can be a difficult task. Server scalability is also a concern with client/server systems. The servers can commonly deal with a large number of users, but there still exists a threshold. Servers can only maintain a limited number of concurrent users because of limitations on memory, CPU power, etc.

Another large concern is the distribution of processing power. The splitting of processing functionality between the server and the client is largely an art. The proper balance of load between thc scrver and the client can determine the success or failure of a system. This debate brings in the discussion of fat server vs. fat client. "Fat client" refers to a system which has a bulk of the processing apportioned to the clients. Conversely, "fat server" refers to a system in which much of the application processing is done at the server. Benefits of a fat server include a decreased load on the network because there is less data being passed back and forth for processing, but fat server systems require more memory and other resources on the server than do fat client applications. As a result, given the limitation on memory and the number of concurrent users, fat server systems do not scale as well as fat client systems. A simple fat client system may only require two percent of a database server's capacity per client, whereas a similar system using a fat server approach may require up to five percent of the server's capacity per client [13].

Other drawbacks of the client/server model include the costs of retraining IS personnel needed to support the system. The costs of retraining one information systems employee could reach $15,000 [19]. A further barrier to client/server implementation is the MIS culture itself. Staff trained in legacy system development with mainframes may be resistant to changing to a client/server environment. This resistance is likely to vary with the length of time that a staff member has been using existing tools, as well as with the technology gap between legacy and client/server applications. Personnel that are trained in the use of LAN's and PC's are likely more able to make the transition to the client/server way of thinking. Still further barriers to the adoption of client/server systems include data security threats, system complexity, and the seemingly exaggerated promises of lower costs.

The future of client/server computing seems to rest on the ability to develop standards to help systems to communicate better. This is what is referred to as open systems: systems that work similarly with the same look and feel regardless of the operating system or underlying database management system. If these problems can be solved, global schemas that provide access to widely distributed, heterogeneous data in an efficient, effortless manner can develop. As will be discussed shortly, the dramatically increased use of Web technology may already be outpacing the traditional client/server models in developing such global schemas.

3. World Wide Web Computing

3.1 History

Many system development platforms and approaches have existed for a long time. The list includes mainframes, which have already been discussed, and relational database systems. In 1993, the World Wide Web became accessible commercially and since then the Web has attracted much attention as an application environment. By mid-1995, the Web became the dominant Internet resource as measured by both packet count and volume [1], surpassing File Transfer Protocol (FTP), Gopher, and Telnet.

A key factor in the rapid success of the Web is the standardization of communications between Web servers and Web clients [8]. On the client side, this communication is handled by software called browsers. Presently, the most commonly used browser is Netscape Navigator. The three main standards governing the communications between Web clients and Web servers are URI (Universal Resource Identifiers), HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language). Together these standards help to support uniform Web development; one person's code is readable and easily understood by other developers. More important is the fact that Web programming is independent of which operating system will run the resulting system. This platform independence is even further supported by the development of Java, a platform-independent, object-oriented language which is used to develop Web applications. HTML is very good for developing the user-interface for a Web application, but it is limited when it comes to complex functions such as dynamic form creation, database access, and conducting real-time calculations. Java and its related language Javascript can be used within HTML coding to improve the functionality of a website. (A website refers to a collection of web pages and all related content for a particular organization or company.) An example of extended functionality on a website would be a web page on-line calculator. Simple HTML creates a static page that cannot deliver results based on inputs to a form. With some Javascript code, a page can be designed which performs desired numerical calculations in real-time.

The use of the Internet continues to grow very rapidly. A recent report in Data Communications [21] pointed out that global spending on networking products and services topped $100 billion in 1996, a 21% increase from 1995. The Internet application market grew by 45% in 1996. Revenue from Internet services such as providers and carriers is expected to leap by 67% in 1997. This article reported that of the more than 100 market researchers, analysts, and vendors interviewed, nearly everyone cited interest in the Internet and Intranets as the reason for the huge increase in network expenditures and traffic. Intranets are closed network systems that are internal to an organization and use Web technology. Users also use Web access software to access Intranets. It seems very apparent from these figures that the interest in the Web is by no means a passing fad.

Another trend in Web-based applications is the trend towards making Web servers more easily set up on desktop workstations. There has been some notion that Web servers may become utilities that are distributed as components of operating systems - perhaps in future versions of MS Windows. Web server software essentially handles the HTTP requests sent from browsers operating on client systems. Typical web server software runs in the background on a server and "waits" for HTTP requests. Therefore, the software must be running at all times in order to effectively respond to requests. Just as calculators, scheduling systems, and word processors have become utilities that are distributed with many operating systems, web server software may also become distributed utilities. A popular example of a web server is Microsoft's Internet Information Server (IIS). According to the IIS website [29], this server permits easy access to a number of different databases, provides security and supports existing Internet standards. IIS only runs on the Windows NT operating system. O'Reilly is the distributor of a commercially available Web server called WebSite, which runs on both Windows 95 and Windows NT [30]. One of O'Reilly's main goals is to have a Web server on every desktop. Although this objective may be somewhat unrealistic, it does reflect the trend of Web servers becoming less draining on computing resources, more easily configured and set up, and more widely available across operating systems. If this trend prevails, the market for networking devices will start to switch from traditional network operating systems to those that utilize TCP/IP (Transmission Control Protocol/Internet Protocol), which is the protocol used for Internet communications. TCP/IP is a packet-oriented family of network and application protocols which is the industry standard for Internet communications and enables Web protocols such as HTTP, HTML and URI. In order to facilitate this communication, a computer browser must be running a TCP/IP stack. TCP/IP stack software is distributed freely with many operating systems such as Windows 95.

3.2 Intranets

The ease of use and setup of web servers has precipitated the onset of a new model of application development called Intranets. Intranets are closed networks within an organization which use web technologies. Each client station has browser software such as Netscape Navigator or Microsoft Explorer and the network applications have the same look and feel as do applications found on the Internet. The key distinction is that the Intranet system is not readily accessible to individuals outside of the organization. It is in this environment that the traditional network operating systems such as Novell NetWare will no longer be required, for it will be replaced by the use of TCP/IP stacks and the use of Internet standards. Since this model is platform independent, the workstations can be any type of system (PC's, Macintosh's, etc.) running any onc of a number of operating systems including Windows or UNIX.

3.3 Advantages of Web Computing Over Client/Server

There are some significant advantages to using the Web over client/server as an application environment [25]. The first is the ease of administration and deployment. Web application programs and files are typically stored upon the server and passed to the client browser only when requested. Thus, maintaining and updating a system normally involves updating the files stored on the server only, which is significantly easier than having to update files and programs stored on the client side as well. Speed of development is also another advantage with Web-based development. Modifications to HTML coding are significantly easier than with most other programming languages, which makes it much easier to use feedback to improve the application. Even if the application is in general use, the changes are easy to make. In applications that involve transaction processing, the updates are still significantly easier than under the client/server model, mainly because of the highly centralized nature of the system. Careful planning must still be done to ensure data and transactional integrity is maintained when the changes are implemented.

3.4 Problems With Web Computing

There are some disadvantages with using the Web as well. One is that portions of a form cannot be updated or activated based on inputs from other parts of a form. For example, using regular HTML it is not possible to update a customer invoice table by simply entering a customer number into a field on the form. This is an easy task for most database management systems such as Oracle and Paradox. This Web problem can be overcome by using Java or Javascript applications, but it also increases the complexity of the development.

A major concern with the Web is that it is a stateless environment, which means that every communication between a browser and a Web server is made with no memory of previous communications. Under this "amnesiac model", as a browser retrieves pages for the user working through a multi-page website, every new page essentially greets this "new" browser. This model is sufficient for most pages that exist on the Web because the vast majority of sites consist of static pages. For robust database applications, this lack of memory can be troublesome. An on-line retail site could not function under this model. Maintaining a memory of who is on-line helps a retail site to track accumulated purchases, maintain a check against a credit limit, and conduct a number of other basic, yet critical, functions.

Netscape is advocating a standard known as "cookies" to overcome this statelessness [17]. Cookies act to move the state information from the server to the browser. Cookies are name/value pairings of variables which are stored on the client's hard drive and tracked by the server. A server, when returning an HTTP object (e.g. HTML document) to a client, may also send a cookie which the client will store. Included with the cookie is a listing of URLs for which that cookie is valid. Any future HTTP requests made by the client to servers included in the URL listing will include a transmittal of the current value of the cookie from the client back to the server. In other words, cookies are passed back from the browser to the server whenever the user submits a form, query, or some other server-generated request form and the server's URL is included in the URL listing associated with that cookie. Once the server has received the cookie, it can consult a database containing cookie values to get information on the particular browser. The only drawback to this approach is that it identifies a browsing client as opposed to an individual. Presumably, other security measures such as passwords can identify an individual to help avoid fraud or unauthorized access.

Other challenges to the Web model include the unpredictability and integrity of transactions. Since it is difficult to manage the user base, it is important to design steps into the transaction process which minimize problems with incorrect or fraudulent transactions. The technical problem here is that the tools for maintaining data and transaction integrity are not mature. Minimal levels of transactional "protection" are currently available, but these dataweb tools will have to mature further to overcome this problem.

Yet another concern about Web application development is the lack of standards to allow integration of systems. This should really be of no surprise since this area is so young. There are two levels of concern: management of the network and management of the application software. Network management systems have been developed that monitor networks. For instance, many network operating systems use SNMP (Simple Network Management Protocol) for such monitoring. Software agents facilitate communication between the management station, which acts as the network central control, and the network elements. Development tools are now being prepared to bridge Web servers and browsers to these SNMP monitors [11]. In fact, Microsoft's Internet Information Server includes SNMP to monitor network performance. In this case the server includes the management station and the network elements include the client browsers. The area of managing application software is a little more complex. Many authors are recognizing that a standard application Management Information Base (MIB) will be required to guide the monitoring and analyzing of software applications [22]. A MIB contains information on the application to help analyze performance, inter-application relationships, and basic definition variables that identify key characteristics of the application. These MIB's could help with the development of dataweb applications. Since these applications may be accessed across a very dispersed network, monitoring performance and other aspects of these applications could be a difficult yet necessary endeavor.

3.5 Models for Web Security

A further concern with the Web model to be considered here is security. In order to conceptualize the security problem, two different models of database security design will be discussed. The first is the Woods Hole architecture [4]. This conceptual model is depicted in Figure 7. (To clarify these diagrams, “untrusted” refers to the lack of database security measures built in to that level. If a front end or a DBMS is labeled “trusted”, that node in the model is entrusted with “guarding” the database against threats.)

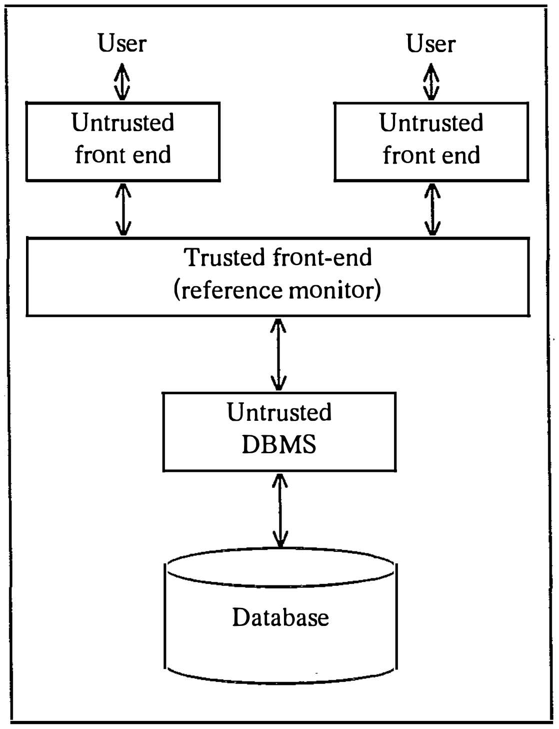

Figure 7. Woods Hole Architecture. From [4]

Under the Woods Hole architecture, the DBMS is untrusted and a trusted front-end lies between the untrusted user and the DBMS. If such an approach were to be utilized in a Web system, there would have to be sophisticated software on the Web server to act as the trusted front end. The problem with this approach is that the tools available at the Web server level are not sophisticated enough at the present time. Tool developers are still trying to develop ways to manage browser statelessness, as previously discussed. Setting up a trusted front-end is significantly more sophisticated than simply managing statelessness. Therefore, this model would not be appropriate for current Web applications. Commercial DBMSs that use variations of this model for their security model include Oracle.

A more appropriate security conceptual model for Web applications is Trusted Subject Architecture [4]. This model is shown in Figure 8. Under this scheme, the need for the trusted

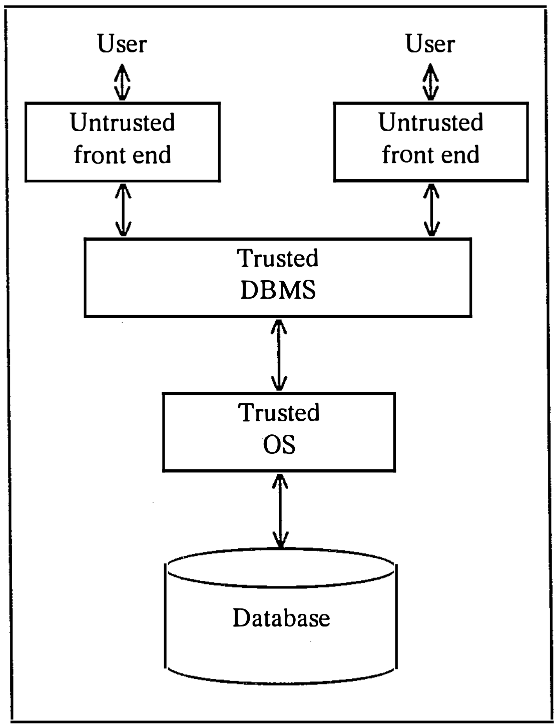

Figure 8. Trusted Subject Architecture. From [4]

front-end intermediary is eliminated because the database and the underlying OS are trusted. With this approach, the DBMS itself monitors the security threats and acts accordingly. This model is more appropriate for the Web. Since there is not enough sophistication at the Web server level to monitor security, it is clear that this function must rest with the DBMS. Most

DBMS's are suited for guarding against security threats. An example of a commercially available DBMS which uses this scheme is Sybase.

These various challenges demonstrate the need for careful planning with respect to Web site or application development. Even though the various programming tools such as HTML, Java, and Common Gateway Interface (CGI) are not too difficult to learn, knowledge of the tools does not ensure a successful website. (CGI is a programming standard which makes it possible for a browser user to initiate and pass information to server-based programs.) A wide variety of technical and business issues need to be addressed in order to overcome some of the potential pitfalls with dataweb development.

Although there is recognized need for standards to help guide the development of Web applications, many vendors are releasing dataweb tools for creating such applications. Thus, robust Web applications are starting to appear and the core technology behind these developments is the database/Web integration. The rest of this paper will focus on issues which pertain specifically to this integration.

4. Integrating the Web and the Client/Server Model

4.1 Overview

As mentioned in the introduction, dataweb is a term used to describe systems which integrate the Web with databases. Datawebs have just started to be developed over the past few years. Although the history of these systems only spans a short period of time, there have been considerable advances. Figure 9 reflects this brief history.

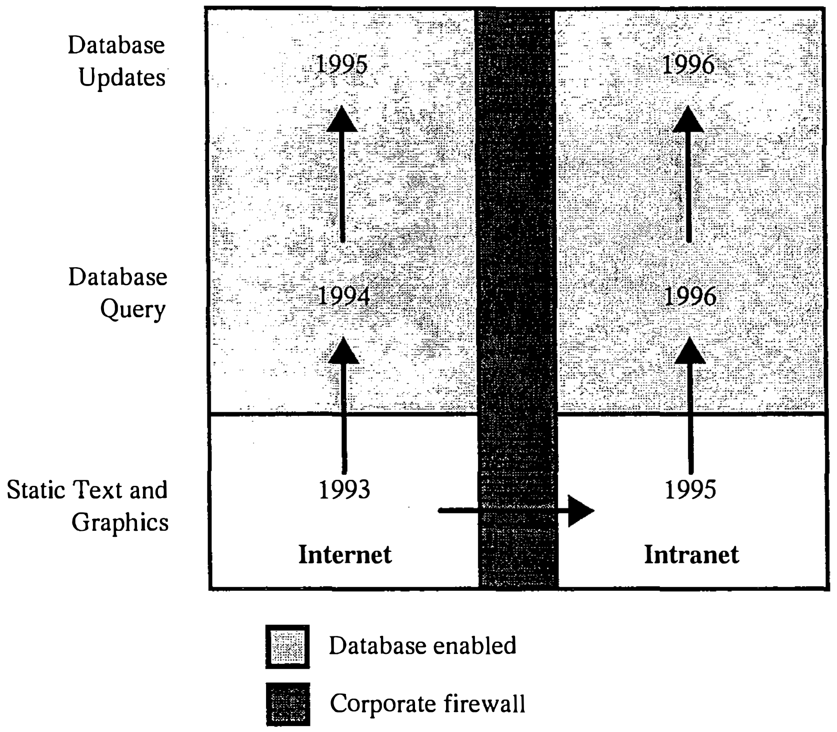

Figure 9. History of DataWebs. Taken from [23]

The Web as a commercial tool appeared in 1993, with static web text and graphics. The technology has improved since then and Web sites are appearing which allow database updates and querying. Generally there are two approaches for updating and querying databases through the Web [8]. The first approach involved using CGI to access databases directly (figure 10). CGI permits the dynamic generation of HTML code. This is significant because it allows Web pages to be generated which are based on query results. The short-coming of this method of database access is that CGI code has to be reprogrammed and recompiled every time a new database access method is required. Furthermore, the CGI code is very specific to the table and access method being addressed.

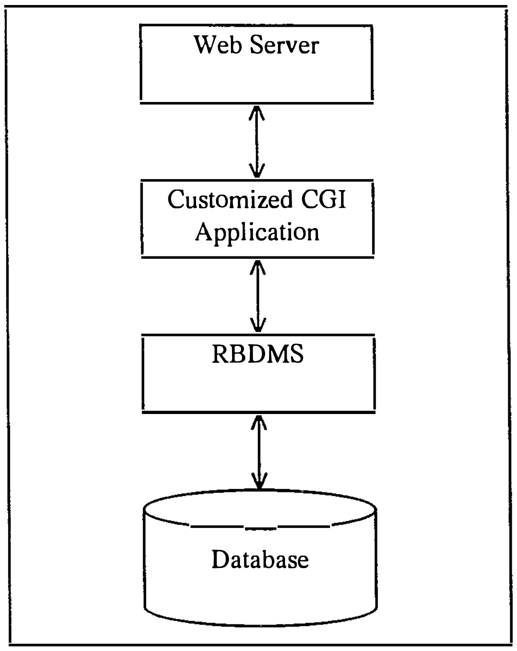

Figure 10. CGI Direct Access to Database (Taken from [8])

The second approach for updating and querying databases over the Web involves using middleware between the server and the underlying database. This model is shown in figure 11. The advantage of this method is that much of the CGI programming is embedded in available functions in the middleware. The disadvantage is the increased overhead of having a software layer situated between the Web server and the database.

Figure 11 also shows the common means by which middleware accesses data from the underlying database. The middleware could be tightly linked to a particular DBMS, such as Oracle, and may access it directly. More commonly, data is accessed by using ODBC (open database connectivity) drivers. ODBC was initially designed by Microsoft to present a common access method for all database and database-related products. The advantage of such drivers is that a system can be set up with an ODBC interface, and everything that supports ODBC will work with it. Virtually all current database products have ODBC drivers. The problem with ODBC is that it addresses data access only from databases. In August 1996, Microsoft released version 1.0 of OLE DB, which specifies a standard method of accessing data from any "data provider", such as an e-mail package, a word processor, or a database. [3]

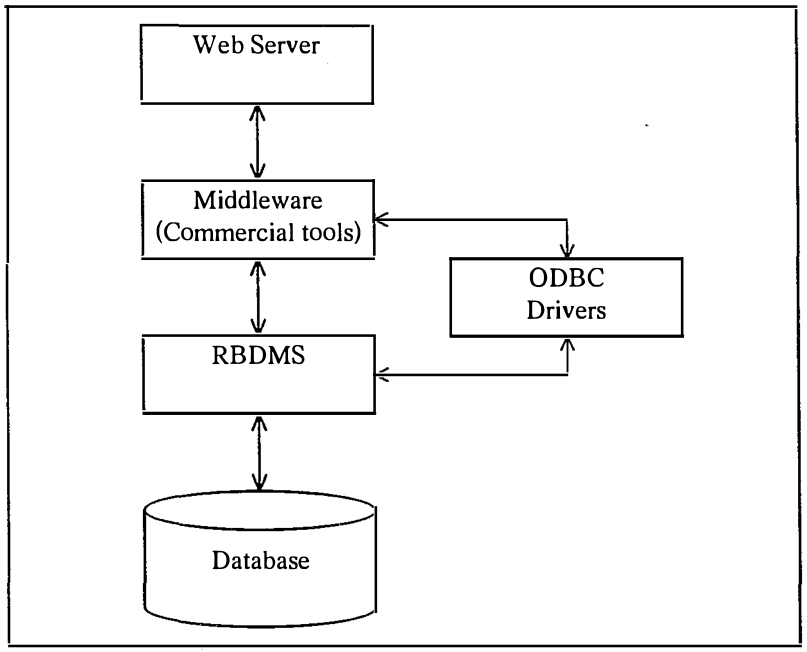

Figure 11. Web Database Access Using Middleware (Taken from [8])

An example of a commercially available middleware package is Cold Fusion from Allaire [27]. The syntax used is called CFML (Cold Fusion Markup Language) and has the same tagged text format as used with HTML. (The syntax was called DBML, database markup language, in versions of Cold Fusion prior to version 2.0) Thus, it is easily adopted by experienced HTML programmers. The key to its functionality is its use of templates, which consist of combined HTML and CFML code. Variables are used in the templates which handle data that is returned by the database query. The result is a valid HTML document that is dynamically generated based upon the data returned from the database. Like Web server software, Cold Fusion must be executing at all times on the server in order to be used. As mentioned above, this results in increased overhead on the server.

Since 1995 there has been an increasing interest in Intranets. Companies are beginning to see significant advantages with setting up internal systems using Web technology. One of the key advances which make Intranets attractive is the development of dataweb integration tools, such as the aforementioned Cold Fusion. There are a number of other similar tools such as InterBuilder by Borland [28]. These tools are based on the middleware model discussed earlier and shown in Figure 11.

These dataweb tools are very important because they help to develop applications which bring together the client/server model with the Web application environment. This integration enhances some of the advantages of each of these two models previously discussed [7]. First, it is a quick development scheme. Since the code is largely HTML, the ease of coding with such a straightforward language is maintained. Also, the deployment costs are significantly lower and the deployment is almost instantaneous. These advantages are due to the highly centralized nature of Web development; all of the code is stored and maintained at the server.

4.2 Thin Client Model for Datawebs

Integrated Web/database systems provide a natural three-tier architecture which introduces a foundation for scalability [16]. The three tiers are the server, the client and the middleware. This facilitates easy addition of clients, since clients only need browser software and the necessary security information (such as passwords) to enter the system.

Many of the current middleware packages come with ODBC drivers to access a large number of different database types. For instance, Cold Fusion comes with drivers to access Sybase, Oracle, MS Access, Paradox, and dBase database files. Hence, none of the underlying data models need to be changed in order to allow integration with the Web. Therefore, operational data stored in existing databases need not be changed and the systems skills of the IT staff can still be utilized, making the transition to datawebs easier for existing staff. Typically, the middleware software is more tightly associated with the server; therefore, it is common to have the middleware run on the server.

The aforementioned problem of increased overhead on the server becomes even more prevalent when discussing the Web as an application platform. Increasing the load on the network by developing fat server systems is more pronounced on the Web, especially during peak times of Web usage. On the other hand, fat client systems, as they are traditionally thought of in the client/server model, may not be very practical because it may be difficult to identify who all the clients are going to be. Since the Web is an open system, users could use browsers from any client to access a Web application. If the application is open to the entire Web such as the case with an on-line retail operation, it is not possible to identify all clients. Therefore, much of the system processing has to be done on the server.

The Web client/server model will only resemble existing client server/models in that the processing power of an application will be distributed between the server and the client. The method of this distribution will vary significantly. Since the users may not even be known in advance it may not be possible to distribute the client side software before the user uses the system. This can be overcome in two ways: one is to explicitly instruct the client to download certain software before using the system; the other is to send packets of processing code to the client as needed. The first method has the advantage of using network bandwidth only once to download software which is then stored on the client. This approach results in a system that is more akin to the client/server model whereby only data and requests for data are passed back and forth between the client and the server. The second approach has broader appeal for the Web, because it eliminates any setup time or effort from the client. The user may not even be aware of whether the client or the server is supporting the processing load. The difference between this approach and earlier client/server systems is that data and associated functions (called methods in object-oriented programming) are being passed to the client, not just data. In the client/server model the client already has the necessary coded functions to process the data. Thus, only data needs to be passed to the client. With datawebs, all of the code resides on the server. Therefore, if the client is to do any processing, it must be supplied the necessary code from the server. Thus, there will more bandwidth required with the second form of the dataweb model. Even with this drawback, this Web client/server model is more appealing than the traditional client/server model because it more automatically deals with previous client/server problems such as version control and software distribution. In addition, programming tools such as Java already exist to facilitate the distribution of segments or granules of code.

An excellent example of a successful dataweb project involves Fruit of the Loom [2]. Fruit of the Loom wanted to increase distributor loyalty to increase its sales, so it decided to launch a Web-based system for order placement and procurement. In order to make the system appealing to its distributors, Fruit of the Loom offered to host its distributors' websites and produce electronic catalogs for these distributors to facilitate order placement. The first distributor to go on-line with the system reached 200 orders per day within a month, with a cost savings of at least $10 per order by eliminating the need for phone-in order staff. This first distributor was up and running within 90 days of the project startup.

5. Accessing Different Types of Databases Over the Web

The first wave of website development involved the construction of fixed files with static text and graphics. Even today, the large majority of websites have this fixed file construction. Although this approach is fine for displaying pre-determined data/text in fixed formats, it only fosters very limited interaction between the client and the server. Not until datawebs started to appear did applications encourage or require interactivity. The rest of this section will briefly discuss three database models and how they fit into the dataweb model. The three are relational, object-oriented and multidimensional models.

The relational model is the basis for most of the dataweb tools available today, since the majority of business applications in use today are based on the relational model and extensions of these applications are natural. Also, many of the applications on the Web are on-line transaction processing (OLTP) systems which primarily use relational database management systems (RDBMSs).

Object databases (ODB's) and relational databases (RDB's) are complementary technologies. RDB's are good choices for systems that support applications with fairly simple data, simple data relationships and static data schema. On the other hand, for systems with complex data and evolving data schema, ODB's are a better choice. ODB's are useful for both content management and providing an infrastructure to increase the value of the Web itself [15]. With respect to content management, objects such as Java applets, complex graphical and spatial layouts or time-series data are much more easily stored and manipulated in ODB's than in RDB's. Also, an object database can act as an intermediary to cache data from a relational database and store the data to allow faster access to pertinent data for a client. The key to this last step is that ODB's can store any analyses of this cached data persistently, whereas information cached by Java is not persistent, regardless of where the data is retrieved. With respect to adding to the infrastructure of the Web or at least a particular website, ODB's could be used to store frequently accessed information, thereby eliminating the need to seek out that information every time it is requested. This information could be variable in context, which could be handled naturally by a ODB. Regardless of the uses for object databases, it is obvious these types of databases can be utilized to improve the functionality of datawebs.

The final database type to consider is the multidimensional model. This model emerged from the need to analyze data from a number of different viewpoints or dimensions. For instance, if a manager is reviewing a summary of sales data, the manager might want to see a breakdown of the data from a number of different views: by date, by product line, by different locations, and/or by salesperson. A properly flexible decision support system will not confine the manager to a predefined pattern of analysis. A decision support system (DSS) is a system that allows analysis and views of data to help managers make decisions. An important element in a DSS is a library of functions and models available to the analyst. Since the user guides the decision process and the system is just a support aid, the system should be very flexible to allow the user a sufficient array of choices to analyze the data. That is, the system should allow managers access to whichever dimension(s) they want to analyze and in whatever order chosen. These types of systems are commonly referred to as On Line Analytic Processing (OLAP) systems. In general, the relational model is not well suited for OLAP [6]. First, RDBs do not consolidate data well across a number of different dimensions. Second, in order to convert a normalized relational database to one which can be analyzed more readily, a large number of denormalized, redundant tables have to generated from existing data. This can cause a significant drain on system resources. In order to avoid these pitfalls, OLAP servers tend to use multidimensional databases to store data and relationships between data. The structure and schema of multidimensional databases are significantly more complex than relational systems, but the data structures allow easier analysis of the data along any dimension or field.

A modification of the multidimensional approach is to employ a RDBMS using a multidimensional database design - a Relational Dimensional Model (RDM) [9]. Information Advantage Inc. is a company that develops and markets DecisionSuite, a DSS software toolset which uses a RDM approach. An extension to DecisionSuite called WebOLAP permits OLAP analysis of structures data over the Web.

As mentioned earlier, most dataweb systems on the Web are OLTP. As the need for data analysis increases, more decision support systems will be implemented using Web technologies, likely within Intranets. At that time, there will be a heightened need to set up OLAP systems over the Web. The largest problem with setting up multidimensional databases on the web is how to present data in various dimensions over the Web. Arbor Corp., designer and distributor of Essbase, a multidimensional decision support aid, has recently announced a Web-based version of Essbase [26]. The results of browser requests are returned as linked Web pages. The problem with this approach is that there could be significant time lags between browser requests. This is especially true with multidimensional systems because the server really has no basis for estimating what the user is going to request next.

As can be seen from discussion, all types of database models will likely be used in robust dataweb applications. What is important is recognizing which one is appropriate under any given situation.

6. Key Features Required for Datawebs

As discussed in [14], there are 6 key business rules which dictate changes to both the server and the client components of an application.

1. Few users make use of all functionality of an application and at any one time even sophisticated users will only use a small portion of an application. Therefore, functionality should be delivered in granules. This parallels the distributed objects client/server model in which large programs are broken down into more manageable components.
2. Application delivery should be adaptable to the different ways users could access the system: different Internet browsers, for instance. Adherence to standards such as HTTP, URI, and HTML is crucial.
3. Data security should be very sophisticated so that access to sensitive information can be restricted.

4. Sophisticated workflow management is required because business processes can be very complex, involving a number of human and system participants. This problem is similar to that encountered on traditional client/server systems.
5. Business data must be delivered automatically, securely and rapidly.
6. The Web interface should reflect the fact that business is task-driven and not concerned with programs or files. In other words, the interface should not include references to file names or programming logic. It should only depict the necessary business processes. For example, an on-line retail system should only guide the user through the necessary tasks involved with making a purchase.

These points offer guiding principles upon which datawebs could be built. A seventh point which was not included in the above list from [14] is that these systems are event-driven and not calendar-driven. For example, instead of delivering hard copies of corporate policies to all departments once a year (calendar-driven), everyone will have access to these changes as they occur if these policies are on the Web (event-driven).

7. On-line Entrepreneurship Bibliography: An Applied Dataweb Example

A bibliography of reference material that is relevant for entrepreneurship in Canada has been developed and maintained by the Innovation Research Centre (IRC), located in McMaster's business faculty. In effort to make this bibliography more broadly available, it was decided to place it on the Web. This section discusses this dataweb.

The first step was to decide which dataweb model would be most appropriate. It was decided to follow the middleware model with Allaire's Cold Fusion as the middleware tool and Borland's Paradox as the DBMS (Figure 12).

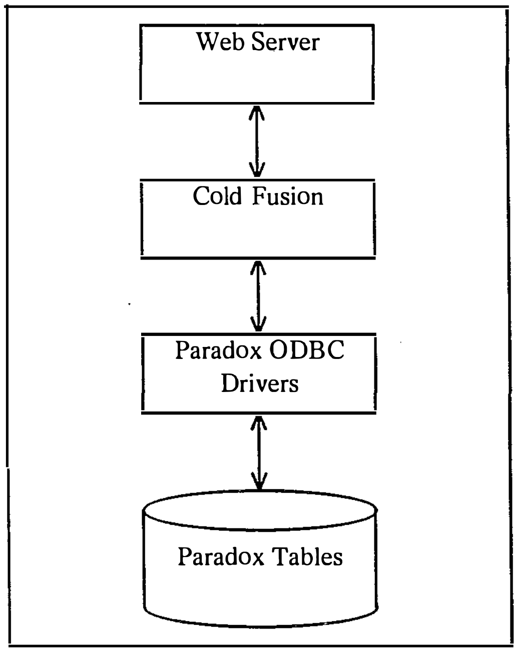

Figure 12. Paradox/Cold Fusion Model Chosen for Project

This combination of model and tool was chosen for a number of reasons. First, the data was initially stored in a format which was easily exported into Paradox tables. Second, upon review of a number of different dataweb development tools, Cold Fusion was chosen because of its apparent ease of use and powerful versatility. As already described, Cold Fusion uses a mark-up language similar to HTML which makes it easier to learn as compared to other choices such as CGI scripting. Cold Fusion is shipped with many ODBC drivers for access to a wide number of DBMS's: Oracle, Sybase, Access, dBase and Paradox to name a few. Also, by choosing a middleware approach, database storage, user interface, and the underlying processing logic were all handled separately. This made the system much easier to design and construct.

The resulting system consists of two modules. The first module permits users to query the bibliography and enter new items to the system. These steps are all done on-line. The submodule that allows users to enter new items presented a peculiar problem. When a user actually enters an item, the item is stored in a temporary table for the project manager to edit before it is entered into the main table. Since most users likely will be infrequent contributors to the database, the question arose as to how much on-line help to assist the user in entering data as correctly as possible (so as to minimize further editing). If little guidance were supplied, the project manager would have to spend a lot of time editing, which was undesirable. If too much help were offered, the Web interface could quickly become full of help information which could confuse or insult a user. A balance was struck between these two extremes, but further use of the system will dictate whether or not modifications should be made. The second part of this module is the querying tool. This module demonstrates the ease with which Cold Fusion can be used to develop drill-down data inquiry applications. Users are first shown a summary of items which match their query. The user then has the choice of clicking a button to obtain more detailed information on any of the items.

A further strength of Cold Fusion was demonstrated in the construction of the query system. It was decided to leave the underlying Paradox table unnormalized. Some of the input items on the Web interface include check-boxes which could have any number of the choices selected. For instance, the "Subject" field has 16 selections from which to choose. The user could select any number of them, from none to 16. If the user selects any of the choices before submitting the form, a variable called "Subjects" is passed to Cold Fusion. The value of this variable is a text string consisting of all subjects selected delimited with commas. For ease of data manipulation, it was decided to store this "Subjects" variable in one field in the table. This results in a table that is not normalized. Using carefully constructed combinations of available commands, Cold Fusion was easily able to handle data manipulation of this unnormalized table.

The second module of this bibliographic system is the data maintenance module. In this module, the project manager can edit or delete items from both the temporary and permanent tables, add items to the temporary table, and move items from the temporary to the permanent table. Thus, all necessary data maintenance functions are available to the manager through the Web. The reason why this approach was chosen was to allow the manager to edit items from anywhere with a Web browser. This obviously brought up a security issue. It was decided to use "cookies" along with a password system to make the system secure. The password screen is the first page that a user sees upon entering this module. Once the user enters the password and user ID, the system verifies the information. If the password information is incorrect, the user is returned to the password input screen. If the information is correct, the user is permitted entry into the system. Also, the server returns a Cookie to the client browser that the system uses in the background to make sure the user is valid. This validity check is done every time a new page is requested by the browser. This helps to ensure that an intruder cannot simply by-pass the password screen if they know the URLs of the site.

Since this project was undertaken in a university, it was logical to turn this tool into a teaching aid also. In order to allow students to browse the maintenance module to get a better idea of the capabilities of Cold Fusion, using "Guest" as both the user ID and password allows people to enter the module. The users can then see everything that the project manager sees and can also complete data forms. The key difference is that any changes that are attempted are rejected and a friendly note comes up telling the user what changes would have been made if they were not simply guests. The security of this system demonstrated the ability of Cold Fusion to use cookies to overcome HTTP statelessness and maintain information on the users.

Cold Fusion is a powerful yet straightforward tool to use for dataweb development. It is a middleware tool that is positioned between the Web interface and the underlying DBMS. This on-line bibliographic system is a good demonstration of the capabilities of Cold Fusion in a dataweb application.

To access the on-line bibliography system:

Query/Input Module:	http://irc.mcmaster.ca/annbib/mainbib.htm
Maintenance Module:	http://irc.mcmaster.ca/annbib/fixbib.htm

Use "Guest" (without quotes) as both the user ID and password.

8. Summary

Database and Web integration are the driving force behind the new dynamic datawebs. Both legacy databases and newly developed systems can be integrated easily into datawebs. There is a significant savings in development time with these systems as compared to traditional client/server systems, but there are a few drawbacks which require more research. Some of these include how to automatically provide data, how to minimize bandwidth requirements, how to best present results of complex queries, and how to improve data integrity and security. These datawebs may be new, but they represent the future in application development. The next wave of website development will be full blown applications built from the ground up. The success of these systems will depend entirely on the lessons being learned today.

Future research ideas include how to best present data to the user. For instance [5] discusses visual interaction with database queries. These systems utilize visual query languages (VQLs) which present the user with more intuitive querying devices such as sliders and toggles. Using 2D and 3D graphs to represent the database, users can click on a plotted point or cube to browse the database. This is just one possibility for portraying both unstructured and structured data to a user. The focus of future research here includes determining effective visualizations of multimedia information.

Researchers could also investigate the relative effectiveness of the various data models in datawebs. This is especially important for web OLAP systems. Are multidimensional databases appropriate or are relational OLAP systems adequate? Under what circumstances are the various models more appropriate? What features of the underlying systems make one data model more appropriate than others? Other research ideas could involve the security issues surrounding dataweb integration. This is still a fairly new application area, but the high degree of interest it has already attracted will stimulate additional research.

9. References

1. Berghel, H., "The client's side of the world wide web", Communications of the ACM, vol. 39, no. 1, January 1996, pp 30-40.

2. Black, B., "Database and the web: giant steps", Database Programming and Design, vol. 10, no. 1, January 1997, pp 34-41.

3. Campbell, R. "OLE DB - moving ODBC into the future", Databased Advisor, vol. 15, no. 1, January 1997, pp. 36-37.

4. Castano, S., Fugini, M., Martella, G., and Samarati, P., Database Security, Wokingham: Addison-Wesley, 1995.

5. Catarci, T., "Interaction with databases", IEEE Computer Graphics and Applications, vol. 16, no. 2, March 1996, pp 67-69.

6. Finkelstein, R., "Understanding the need for on-line analytic servers", <http://www.arborsoft.com/essbase/wht_ppr/finkTOC.html> December 10, 1996.

7. Furey, T., "The web changes the client/server game", <http://www.csc.ibm.com/advisor/library/2c4a_135a.html> Feb. 6, 1997.

8. Hadjiefthymiades, S. and Martakos, D., "A generic framework for the deployment of structured databases on the world wide web", Computer Networks and ISDN Systems, vol. 28, iss., 1996, 1-2, pp. 1139-1148.

9. Information Advantage, "OLAP - scaling to the masses", Report ordered through the Web received through mail directly from company.<http://www.infoadvan.com> January 24, 1997.

10. Information Advantage, "The impact of decision support system architecture", Report ordered through the Web received through mail directly from company. <http://www.infoadvan.com> January 24, 1997.

11. Jander, M., "Welcome to the revolution", <http://www.data.com/roundups/monitor.html> December 2, 1996.

12. Lazur, B. "Transaction processing monitors meet the intranet", ZD Internet, vol. 2, iss. 3, March 1997, pp. 99-104.

13. Linthicum, D., "C/S collapse", <http://www.dbmsmag.com/9612d07.html> December 2, 1996.

14. McKie, S., "Internet-DBMS strategies", <http:// www.dbmsmag.com/9610d13.html>, October 16, 1996.

15. Mellman, J., "Object databases on the web", <http://www.webtechniques.com/features/sep96/mellman.shtm> October 23, 1996.

16. Mixnet, "Delivering internet/intranet applications in web days instead of man years", <http://www.mixnet.com/database.htm>, November 7, 1996.

17. Online Development, "Database integration and the world wide web", <http://www.ondev.com/spmi/database.html>, June 1, 1996.

18. Orfali, R., Harkey, D., and Edwards, J. "Intergalactic client/server computing", Byte, vol 20. April 1995, pp. 108-110, 114, 116, 120, 122.

19. Schultheis, R. and Bock, D., "Benefits and barriers to client/server computing", IS Management, vol. 45, Feb 1994, pp. 12-15, 39-41.

20. Simon, A., Strategic Database Technology: Management for the Year 2000, San Francisco: Morgan Kaufman (1995).

21. Staff of Data Communications, "The 1997 data comm market forecast", <http://www.dtat.com/roundups/forecast97.html> December 3, 1996.

22. Sturm, R. and Weinstock, J., "Application MIBs: taming the software beast", <http://www.data.com/Tutorials/Application_MIBs.html> November28, 1996.

23. Thomson, D., "Corporate developer: web-footed applications", Database Programming and Design, (August 1996), pp. 63-67.

24. Watterson, K. Client Server Technology for Managers, Reading: Addison-Wesley (1995).

25. Whetzel, J., "Integrating the world wide web and database technology", AT&T Technical Journal, vol. 75, iss. 2, pp. 38-46.

26. <http://webgate.arborsoft.com/tbc/home.htm>

27. <http://www.allaire.com>

28. <http://www.borland.com>

29. <http://www.microsoft.com/iis/>

30. <http://ora.com/catalog/webpro/>

INNOVATION RESEARCH WORKING GROUP
WORKING PAPER SERIES

1. R.G. Cooper and E.J. Kleinschmidt, "How the New Product Impacts on Success and Failure in the Chemical Industry", February, 1992.

2. R.G. Cooper and E.J. Kleinschmidt, "Major New Products: What Distinguishes the Winners in the Chemical Industry", February, 1992.

3. J. Miltenburg, "On the Equivalence of JIT and MRP as Technologies for Reducing Wastes in Manufacturing, March, 1992.

4. J.B. Kim, I. Krinsky and J. Lee, "Valuation of Initial Public Offerings: Evidence from Korea", February, 1992.

5. M. Basadur and S. Robinson, "The New Creative Thinking Skills Needed for Total Quality Management to Become Fact, Not Just Philosophy", April, 1992.

6. S. Edgett and S. Parkinson, "The Development of New Services Distinguishing Between Success and Failure", April, 1992.

7. A.R. Montazemi and K.M. Gupta, "Planning and Development of Information Systems Towards Strategic Advantage of a Firm", April, 1992.

8. A.R. Montazemi, "Reducing the Complexity of MIS Innovation Through Hypermedia and Expert Systems", May, 1992.

9. M. Basadur and Bruce Paton, "Creativity Boosts Profits in Recessionary Times - Broadening the Playing Field", June, 1992.

10. Robert G. Cooper and Elko Kleinschmidt, "Stage-Gate Systems for Product Innovation: Rationale and Results", June, 1992.

11. S.A.W. Drew, "The Strategic Management of Innovation in the Financial Services Industry: An Empirical Study", July, 1992.

12. M. Shehata and M.E. Ibrahim, "The Impact of Tax Policies on Firms' R & D Spending Behavior: The Case of R & D Tax Credit", July, 1992.

13. Willi H. Wiesner, "Development Interview Technology: Implications for Innovative Organizations", July, 1992.

14. Isik U. Zeytinoglu, "Technological Innovation and the Creation of a New Type of Employment: Telework", August, 1992.

15. John W. Medcof, "An Integrated Model for Teaching the Management of Innovation in the Introduction to Organizational Behaviour Course", October, 1992.

16. Min Basadur, "The Why-What's Stopping Analysis: A New Methodology for Formulating Ill-Structured Problems", October, 1992.

17. Stephen A.W. Drew, "Strategy, Innovation and Organizational Learning an Integrative Framework, Case Histories and Directions for Research", November, 1992.

18. Stephen A.W. Drew, "Innovation and Strategy in Financial Services", November, 1992.

19. Scott Edgett, "New Product Development Practices for Retail Financial Services", November, 1992.

20. Robert G. Cooper and Elko J. Kleinschmidt, "New Product Winners and Losers: The Relative Importance of Success Factors - Perception vs. Reality", November, 1992.

21. Robert G. Cooper and Elko J. Kleinschmidt, "A New Product Success Factors Model: An Empirical Validation", November, 1992.

22. Robert G. Cooper & Elko J. Kleinschmidt, "Stage Gate Systems: A Game Plan for New Product Success", November, 1992.

23. Min Basadur, "Optimal Ideation-Evaluation Ratios", March, 1993.

24. Christopher K. Bart, "Gagging on Chaos", March, 1993.

25. Yufei Yuan, "The Role of Information Technology in Business Innovation", July, 1993.

26. Isik Urla Zeytinoglu, "Innovation in Employment: A Telework Experiment in Ontario", July, 1993.

27. John Miltenburg and David Sparling, "Managing and Reducing Total Cycle Time: Models and Analysis", August, 1993.

28. R.G. Cooper, C.J. Easingwood, S. Edgett, E.J. Kleinschmidt and C. Storey, "What Distinguishes the Top Performers in Financial Services", September, 1993.

29. B.E. Lynn, "Innovation and Accounting Research", September, 1993.

30. Min Basadur and Peter Hausdorf, "Measuring Additional Divergent Thinking Attitudes Related to Creative Problem Solving and Innovation Management", November, 1993.

31. R.G. Cooper and E.J. Kleinschmidt, "Determinants of Time Efficiency in Product Development", December, 1993.

32. Christopher K. Bart, "Back to the Future: Timeless Lessons for Organizational Success", February, 1994.

33. Ken R. Deal and Scott J. Edgett, "Determining Success Criteria for New Financial Products; A Comparative Analysis of CART, Logit and Discriminant Analysis", February, 1995.

34. Christopher K. Bart and Mark C. Baetz, "Does Mission Matter?", February, 1995.

35. Christopher K. Bart, "Controlling New Products: A Contingency Approach", February, 1995.

36. Christopher K. Bart, "Is Fortune Magazine Right? An Investigation into the Application of Deutschman's 16 High-Tech Management Practices", February, 1995.

37. Christopher K. Bart, "The Impact of Mission on Firm Innovativeness", February, 1995.

38. John W. Medcof, "Transnational Technology Networks", April, 1995.

39. R.G. Cooper and E.J. Kleinschmidt, "Benchmarking the Critical Success Factors of Firms' New Product Development Programs", April, 1995.

40. John W. Medcof, "Trends in Selected High Technology Industries", July, 1995.

41. Robert C. Cooper & E.J. Kleinschmidt, "Benchmarking Firms' New Product Performance & Practices", September, 1995.

42. Min Basadur and Darryl Kirkland, "Training Effects on the Divergent Thinking Attitudes of South American Managers", November, 1995.

43. Min Basadur, "Organizational Development Interventions for Enhancing Creativity in the Workplace", November, 1995.

44. Min Basadur, "Training Managerial Evaluative and Ideational Skills in Creative Problem Solving: A Causal Model", December, 1995.

45. Min Basadur, Pam Pringle and Simon Taggar, "Improving the Reliability of Three New Scales Which Measure Three New Divergent Thinking Attitudes Related to Organizational Creativity", December, 1995.

46. N. P. Archer and F. Ghasemzadeh, "Project Portfolio Selection Techniques: A Review and a Suggested Integrated Approach", February, 1996.

47. Elko J. Kleinschmidt, "Successful new product development in Australia: An empirical analysis", February, 1996.

48. Christopher K. Bart, "Industrial Firms & the Power of Mission," April, 1996.

49. N. P. Archer and F. Ghasemzadeh, "Project Portfolio Selection Management through Decision Support: A System Prototype," April, 1996.

50. John W. Medcof, "Challenges in Collaboration Management in Overseas Technology Units," April, 1996.

51. Susan L. Kichuk and Willi H. Wiesner, "Personality and Team Performance: Implications for Selecting Successful Product Design Teams," May, 1996.

52. Susan L. Kichuk and Willi H. Wiesner, "Selection Measures for a Team Environment: The Relationships among the Wonderlic Personnel Test, The Neo-FFI, and the Teamwork KSA Test, " May, 1996.

53. Susan L. Kichuk and Willi H. Wiesner, "Personality, Performance, Satisfaction, and Potential Longevity in Product Design Teams," June, 1996.

54. John W. Medcof, "Learning, Positioning and Alliance Partner Selection," June, 1996.

55. Scott J. Edgett, "The New Product Development Process for Commercial Financial Services," July, 1996.

56. Christopher K. Bart, "Sex Lies & Mission Statements," September, 1996.

57. Stuart Mestelman and Mohamed Shehata, "The Impact of Research and Development Subsidies on the Employment of Research and Development Inputs," November, 1966.

58. Mark C. Baetz and Christopher K. Bart, "Developing Mission Statements Which Work," November, 1996.

59. Fereidoun Ghasemzadeh, Norm Archer and Paul Iyogun, "A Zero-One Model for Project Portfolio Selection and Scheduling," December, 1996.

60. R. G. Cooper, S. J. Edgett, E. J. Kleinschmidt, "Portfolio Management in New Product Development: Lessons from Leading Firms," February 1997.

61. R. G. Cooper, S. J. Edgett, E. J. Kleinschmidt, "Portfolio Management in New Product Development: Lessons from Leading Firms -- Part II," February 1997.

62. C. K. Bart, "A Comparison of Mission Statements & Their Rationales in Innovative and Non-Innovative Firms," February 1997.

Innis
REF
HD
45
.W657
no. 63