REAL-TIME COMPUTING: MEETING THE CLIENT'S DEMAND FOR DATA ACCESS VIA THE INTERNET

S. Sitharama Iyengar and Brian E. Pangburn

Abstract. With the tremendous growth of the Internet, many companies are feeling pressure from their clients to provide real-time access to data. A flashy, static web site is no longer enough to satisfy those who demand 24-hour access to dynamic data including billing, personal profiles, and financial statements. We will give an overview of the process for putting data online - from analyzing existing shortcomings to methods for Internet database connectivity and developing online calculation engines. Finally, we will discuss some trends in data mining and present ideas for developing a distributed data mining system.

1. Existing Data Formats. Before considering new applications for corporate data, an analysis of current in-house technology is crucial. Many companies do not utilize a structured database. Data files may be in independent ASCII or binary files that can only be utilized by the software that created them. Others may be in spreadsheets or outdated database systems that are not suitable for new technology. In recent years, database systems have been developed that are both powerful and robust. They provide a layer of abstraction that allows for separate management of applications and data. These systems are usually multi-user, secure, scalable, and better suited for recovery. Modern databases also open the door to cutting edge technology in data analysis via data mining, and data communications via Internet connectivity .

Once a decision has been made to upgrade database technology, it is important to restructure data. A careful analysis will reveal existing redundancies and potential for optimization. A good design will take full advantage of relationships that exist among data and allow for growth. For some introductory material on database design, see Date [1].

2. Data Security. When organizing and structuring data into a modern database, security must be considered. Not only is it important to protect a company's internal data, but also to protect the privacy of it's clients. For a small database that will only be used by a select few, it may be enough to add a simple password to the login interface. However, for most enterprises, user-level security and encryption are the minimum. User-level security refers to creating a list of users each of whom has a password which allows access to appropriate components of the database. The actions that a given user can perform on his/her "view" of the database can also be restricted. For example, a systems administrator would have permission to perform any action on any component of the system whereas new data entry personnel might only have permission to read/write certain tables containing less sensitive data. Encryption adds an additional level of security by making data unreadable to applications outside of the database system.

When considering sharing a database outside of a closed network such as on the Internet, other security factors must be considered. First of all, the local network must be protected. Any form of open connection to the Internet is subject to attach by hackers. For example, InsWeb, a company serving the insurance industry, estimates that they receive "at least five or

six serious break-in attempts per day." [2] Firewalls can be implemented to protect a local network by limiting the access privileges of outsiders. The second consideration is the security of data passed outside of the firewall. Any data sent from the client to a server including password information is subject to interception. Additionally, data returned by the server to a client is also at risk. Secure Hypertext Transfer Protocol (https), digital signatures, and encryption keys can all be used to combat piracy of data *en route*. Another option is to employ a third party service to handle access to corporate networks and sensitive data. With security requirements ever changing, an outside specialist may be the safest solution.

3. Client Views and Date Sensitivity. Once general security is in place, the appropriateness of data should be evaluated. It should go without saying that a client should only be allowed access to their own data, but there is also a subjective aspect. Sending a client too much information can be dangerous. For example, a set of internal calculation codes are likely to raise more questions than they answer. Data should be presented in a clear and concise manner so that a client is only presented with what they are seeking. By carefully selecting what data is sent, less bandwidth will be consumed and response times should be faster. As much filtering as possible should be done on the server-side.

A lot of data is also date-sensitive. Especially when dealing with financial statements or billing, software systems may only update a database daily, weekly, monthly, or even less frequently. Clients should probably not be given access to data while it is being calculated or updated. Reports should always be clear about the effective date for a given report. If real-time calculations are being done, there must be verification that a request date is valid.

4. Client and Server Platform Considerations. To allow as many users as possible to retrieve their information online, client and server platforms must be addressed. A server should be able to handle requests from Macintosh, UNIX, and Windows based web browsers at a bare minimum. One of the best ways to accomplish this is by sending all requests and returning all data in HTML format. For example, a client might enter name, password, and request information from their browser. This data can be embedded in HTML statements and passed as a request to the server. The server then interprets the HTML, processes the request, and returns data in HTML format to the client. On the client-side, access speed is also a major consideration. While it is tempting to put together a fancy interface, clients with low connection speeds may quickly become frustrated waiting for a site to load.

On the server-side, the two major considerations are functionality and capacity. If using a third party to host the web site, the outside server must be compatible with the database system. Often a special type of web server is required along with software to communicate with the data files. Not only must the server be compatible, but also the third party must allow for the installation of the associated software. If the site hosting is to be done in-house, then security and bandwidth become the major considerations. Once security is addressed, adequate communication hardware is needed to handle web traffic. Even web sites published on closed networks or "intranets" can face enormous traffic. For example, Chrysler Corp. estimates that their employee Intranet receives between 800,000 and 1 million hits each week. [3] "Popular WWW sites such

as *Lycos* and *AltaVista* receive three to ten million accesses a day." [4] If a site is performing dynamic activities including database transactions, it will require even more bandwidth and can quickly become bogged down by high traffic.

5. Methods of Database Connectivity. The next step in making a database accessible via the Internet is to connect it to a web site. Database connectivity refers to the ability to connect a database to some outside source. Traditionally, this has meant an interface with some programming language. Connecting a database to a remote web browser adds a new level of complexity.

To address the issue of database connectivity as a whole, Microsoft has developed the Universal Data Access (UDA) model. This model provides a system of software components that allow platform independent communications among relational databases, spreadsheets, and indexed-sequential files. Instead of trying to create the ultimate database format to consolidate every possible data type, UDA attempts to create the ultimate interface to unify communication among different data formats. The model consists of a system level interface called OLE DB and an application level interface called ADO. A programming language can interface at either level, but ADO provides a higher level of abstraction for database functionality.

The Remote Data Service (RDS) component of UDA specifically addresses remote data access via the Internet. It supports OLE DB and ADO while allowing for inconsistent or even absent server connections. By shifting a set of data and operations to the client machine, manipulations can be done without a persistent connection. Data can later be uploaded to the server for resynchronization. [5]

For the task of deploying an existing database on the Internet, Microsoft also introduced Active Server Pages (ASP) technology, which is a combination of an ActiveX scripting language and their Active Data Objects (ADO) model. While limiting the choice for a web server, ASP can use pure HTML to communicate with the client – solving the problem of compatibility with the client's web browser. ASP also helps to ease the burden of multiple connections. The server can be instructed to maintain a minimum number of connections to the database and then use those connections as requests are made. This helps to minimize the additional overhead required to initiate and terminate connections. Additionally, ASP works well with the platform independent JAVA. Some of the server constraints can even be overcome thanks to the development of third party ASP clones for other platforms. [6]

Various plug-ins and controls are available that can simplify many connectivity issues, but most are platform dependant and some require additional licensing. There are also other methods including CGI scripting that can be used for database communication, but they are more limited in function.

6. Software Development. Once database, security, platform, and connectivity issues have been addressed, web site software development can be considered. The web interface can make the difference as to whether or not a system provides "value added" to the client. A standard, user-friendly screen is a must and it has to be obvious what the user needs to do with little or no instruction. There is a lot of web site development software on the market now and the biggest problem again becomes platform or browser dependency.

Based on the nature of the data, a decision must be made regarding whether a static or dynamic data file will be used, and if any calculations are to be done on the client or server-side. A static data file is a snapshot of a database that is available for browsing. This is effective for data that does not change very often. The client can view, but not modify data and the entire data file must be re-posted when it is modified on the server-side.

The location for a calculation engine is somewhat subjective and several factors must be considered. If calculations are to be done on the client machine, the user must either download the engine at runtime or obtain the program through some other means. Downloading at runtime insures that the user has the latest version of the code, but platform issues and download time can be major obstacles. The following table lists some of the issues to consider:

Factor	Location of Code
Very Large Calculation Engine	Server
Platform Dependant Code	Server
Aggregate Record Calculations	Server
Large Input Set with Small Output Set	Server
Small Input Set with Large Output Set	Client
Small Output Set	Either
Heavy Load on Server	Client
Distributed Data Sets	Client
Many Iterations / Scenarios	Client

If calculations are going to be kept on the server, the client-side user interface can probably be written in HTML and a scripting language such as CGI or VBScript. However, if a full application with calculations is going to be deployed on the client's machine, software will need to be written. Unless the client base is small and there will be no cross-platforming, JAVA is probably the best solution.

While requiring the developer to have knowledge of object oriented programming, JAVA can be used to develop both stand-alone local applications and Internet applets that are platform independent. JAVA code is compiled into bytecode which can then be downloaded to any machine with a JAVA interpreter. Once loaded onto the user's machine, these applets are secure and capable of database connectivity. Recently, just-in-time compilers (JIT) have been developed for some systems to compile bytecode into native code after download for faster execution. [7]

We will now look at a project currently being developed which will allow client companies to update their employees' retirement contributions, and produce participant benefit statements via the Internet. This project stems from the long turnaround times required to produce client statements without an automated data feed. Traditionally, statements have been generated using the following process:

1. Client's payroll department sends hardcopy of contribution information to retirement plan Administrator
2. Administrator updates contribution and investment performance information and produces benefit statements
3. Statements are mailed back to client and distributed to employees

The process usually takes close to two weeks and puts a tremendous burden on the Administrator's data entry personnel at the end of each quarter.

The system being developed will significantly reduce the burden on data entry personnel and increase turn around time. Using a JAVA applet, the client will be able to:

1. Download the JAVA applet and/or any updates
2. Establish a secure connection to the Administrator's database
3. Enter the desired reporting period
4. Update the contribution information for each employee while the starting balances and investment information for the desired reporting period are downloaded as a background process
5. Calculate the new account balances and create the associated benefit statements

Using this approach, the Administrator remains in full control of the investment performance data while the burden of data entry distributed among all of the clients. The client controls the timing of the statements and distribution to the participants. Since the payout liability falls on the client for these particular types of retirement plans, there is little risk of tampering with contribution amounts. All statements would contain caveats and regular audits of the contribution data would be performed by the Administrator.

7. A Model for Distributed Data Mining. In recent years data mining has come to the forefront of database technology. Moxon defines data mining as "a set of techniques used in an automated approach to exhaustively explore and bring to the surface complex relationships in very large datasets." [8] The basic idea is to deploy a set of algorithms on a datafile and let them analyze the data without user intervention. This shifts the workload to the computer and allows for a level of analysis that would be impossible for a human. Data mining can be applied to any field with a significant bank of data to reveal patterns in everything from weather to purchasing.

Data mining uses techniques including neural nets, decision trees, genetic algorithms, nearest neighbor, and rule induction for data analysis. [9] The traditional approach is to start with a large, centralized data store or warehouse, set up parameters for the desired type of analysis, allow the calculation engine to analyze the data, and then extract the results to a report. Companies are relying on data mining more and more due to the huge increases in stored information. [8,9] With technology like UDA emerging, and increases in distributed data sources, the future of data mining may shift to distributed analysis.

Not only could a single company with distributed data reveal trends in their own data, but also combine data from a wide variety of outside sources for enhanced modeling. Since a single data source may be massive, there is no way that a client-side analysis tool could download all the data for local analysis. Instead, there would probably be multiple levels of analysis and sampling. For example, a request could be sent to several server-side data mining engines to return a particular set of demographical correlations. These results could then be merged and re-mined locally to find trends across the different data sources. Another approach would be to bypass server-side mining and sample the larger datasets so that all analysis could be done locally on merged data subsets.

Today, some companies strip their data files of proprietary or secure information and sell them to solicitors or analysts. In the near future, anyone on the Internet may be able to pay for access time on web sites that link to different data repositories, and utilize local and remote mining tools to perform research. As communication speeds increase and data access evolves, distributed data mining may become as simple as using a search engine.

Conclusion. We have given a broad overview of several areas related to putting

data online. There are many benefits to structuring corporate data including increased security and greater analytical power. Both companies and clients can benefit from real-time, remote data access. Clients can even be enticed into sharing data entry workloads. If filtered carefully, sharing corporate data with those outside of the existing client base may also prove profitable as long as proprietary information is protected.

REFERENCES

[1] Date, C. J. *An Introduction to Database Systems*, 6th ed. Addison-Wesley Publishing Company, Inc., Reading, MA, 1995.

[2] West, Diane. Web Security: Still A Wild And Woolly World. *National Underwriter* (January 26, 1998), pp. 7, 10.

[3] Quinn, Richard D. Net gains keep firms on pace with demand for self-service. *Employee Benefit News* (February 1998), pp. 44-46.

[4] Andresen, D., Yang, T., and Ibarra, O. Toward a Scalable Distributed WWW Server on Workstation Clusters. *Journal of Parallel and Distributed Computing, Academic Press*, 1997.

[5] *OLE DB/ADO: Making Universal Data Access a Reality*, Microsoft Corporation, 1997. Accessible at http://www.microsoft.com/data/ado/sigmod98.htm.

[6] Litwin, Paul. Publishing Active Server Pages. *Smart Access* (August 1997), pp. 1, 3-8.

[7] Morgan, Bryan, et al. *Visual J++ 1.1 UNLEASHED*, 2nd ed. Sams.net Publishing, Indianapolis, IN, 1997.

[8] Moxon, Bruce. Defining Data Mining. *DBMS Data Warehouse Supplement*, August 1996. Accessible at http://www.dbmsmag.com/9608d53.html.

[9] An Introduction to Data Mining, Pilot Software, Inc., 1997. Accessible at http://www.pilotsw.com/dmpaper/dmindex.htm.