Paper 1080

Using The WEB for Data Warehousing

ARCHITECTING A

DATA WAREHOUSE/INTRANET SOLUTION

Hewlett Packard

Presented by

Roger Eberlin
Hewlett Packard
290 Woodcliff Drive
Fairport, New York 14450

(716) 264-4032

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

1. Executive Summary - Background and Opportunity

Two of the most talked-about recent computer trends are the Internet (specifically, the Intranet) and data warehousing.

The Internet in the form of the Web environment, is providing new application access and data distribution channels to an organization's internal and external information consumer (customer) base. It is also providing the ability for applications to link disparate data to support business application requirements. Columnar, textual, and multimedia data forms can be linked into what the user of the web environment sees as one single application.

Companies have jumped into data warehousing assembling multiply sourced pieces of data in order to establish information-based competitive advantage (or avoid competitive disadvantage) in the global-competitive market place of the 1990's. A recently completed IDC study found that this hope of harvesting the value of information is quite realistic with the mean 3 year return on investment (ROI) of data warehousing projects being 401 percent! (Steve Graham, *The Foundations of Wisdom: A Study of the Financial Impact of Data Warehousing*, (Toronto: International Data Corporation, 1996), p.5).

A number of organization and business needs are driving web-browser access to data warehouses. Among these needs are the following:

1) Need to expand warehouse access to much larger groups of knowledge workers

A data warehouse is intended to empower knowledge workers (i.e. personnel who make product, service, or organizational recommendations and decisions) to help them in making faster, better recommendations and decisions. Part of the value of a data warehouse is dependent on the degree to which the information within the warehouse is made available to the organization's knowledge workers. In fact, the IDC study previously cited found that lack of wide-spread usage was one of the reasons behind data warehouses with negative ROI.

Traditional decision support systems in the 1980's were typically used by only 2 percent of all corporate knowledge workers--primarily the financial analysts, marketing analysts, and a few executives (using pre-canned executive information systems). Today, in companies with deployed data warehouses, the percentage of knowledge workers actually using the warehouse has typically risen to around 10 to 25 percent. This represents a huge percentage of workers who are still largely untouched by the benefits of direct warehouse-information access. Forward-looking companies understand this and are actively making plans to roll their data warehouse out to much higher percentages of their knowledge workers.

2) Need to reduce client software installation and maintenance costs

One of the limitations to more widely deploying data warehousing access is the very high cost of installing and maintaining complex client/server software. The cost of installing and (as versions

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

change) updating client software is very expensive. Companies would like to eliminate or minimize these costs.

3) Need to provide hardware/operating system independence

Many, if not most, IT departments would like to be able to deploy desktop software that is independent of client hardware platform and operating system. In spite of the dominance of Windows/Intel platforms, the Macintosh continues to have die-hard supporters in core groups such as graphics departments and some executives. Likewise, the UNIX workstation is definitely still the desktop of choice in most engineering departments. Hardware/operating system independence would give IT departments one less headache.

4) Need to selectively open data warehouses to users outside the corporation

A very interesting new trend is the desire of companies to open up their data warehouse(s) to selected users outside of their enterprise. This trend is primarily being driven by the desire for companies to share certain information with business partners, suppliers, distributors, and major customers.

The marriage of the Internet and Data Warehouse technology areas provides a wide data distribution opportunity using low cost commodity components and highly integrated data on user desk tops. The widened data distribution channel presented by joining these important technologies is irresistible from an organization's marketing perspective.

2. Scope

In this White Paper, we will explore the product requirements which will allow an organization to leverage Internet and Web based technology. We will examine the available architectural approaches and technical trade-offs in connecting a data warehouse to the Intranet. The information presented will additionally be compared to building more "conventional" client-server front-end access applications to the warehouse, showing where this architectural choice continues to show high value.

This document looks at the Web-Enabled Warehouse environment, assuming the following perspective:

1) The merits of and design principles for developing and supporting a Data Warehousing approach to data management and information distribution have been adopted by the organization.
2) The Internet is used as a term meaning - delivery mechanisms encompassed by the physical deployment of internal organization Intranets as well as access to the Word Wide Web.
3) Security is a necessary component in planning for the deployment of Web based applications. Security must not only be addressed by the user's access right and "need to know", but also the sensitivity of the data must be established with regard to its access.
4) The methods used to deploy data warehouse information will be dependent upon user expectations for both performance and volume of results to be delivered.

3. Why Consider a Web Front-End to Your Data Warehouse?

Private corporate Intranet sites offer an opportunity for a new level of collaborative information sharing among decision makers throughout the enterprise. It's no wonder that Intranets are the fastest growing segment of the Web. With the right tools and the right architecture, the data warehouse can be made accessible over the enterprise Intranet, forming the basis for a comprehensive enterprise information infrastructure. Or as Karen Watterson[4] more humorously puts it,

> 'Just when you were beginning to feel pretty smug about your competence in client/server application development, WHAM! The Internet hit. ... Then the boss, or [corporate] board, started making noise about making all of those client/server applications "intranet" applications.'

Neil Raden[2] very succinctly gives the reasons for using the Web to access a data warehouse. He states "the compelling advantages in using the Web for access to a data warehouse are almost the same as those for any other application". Here is a closer look at these advantages [2]

*"**Infrastructure:** Using the Web shifts the burden of platform compatibility to the browser vendors".*

> There are still many client platforms to choose from, and support, for your presentation layer, including Windows 3.x, Windows-95, Windows NT, OS/2, Mac OS and/or UNIX. Plus the clients will be installed on many computer systems at your site with a wide variation in memory, disk space, processor speed, and video capability. Supporting all these platforms through traditional IT and application development takes both time and money. Using a Web browser as the client interface to an application is an attempt to shift the compatibility burden to the Web-browser vendor [such as Microsoft, Netscape, Spyglass, etc.]

*"**Access:** Both employees and customers can easily access the Internet, eliminating the need to extend the corporate network".*

> Just about anyone can find entry onto the Internet and internal networks using a Web browser. This eliminates the need for companies to extend their networks to accommodate all potential users.

*"**Cost:** Web browsers are a fraction of the cost of OLAP [and other] client tools.*"

> In 1996, typical costs for OnLine Analytic Processing (OLAP) tools are $500 to $1000 per seat, and most clients only use a small faction of the functionality that those tools provide. Non-OLAP client interfaces may be cheaper, but require custom programming and related support costs. Web browsers can be obtained for under $100, and sometimes for free. They are downloaded easily from the vendors' Web site, and require minimal customization from there.
>
> There is more to the cost of a web-enabled environment beyond that of the front-end browser. This justification for a low cost user interface has development costs on the back-end, which while centralized should not be trivialized.

*"**Leverage:** The Web browser can be used in every application that provides a Web gateway.*"

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Unlike the custom-crafted interfaces in applications created using Microsoft Visual Basic or PowerSoft's PowerBuilder, the Web browser can be used by every application that provides a Web gateway (a translator between HTML and the database server's API -- we'll look at this more later). Also, with access via the Web comes E-mail, Usenet news, and a host of other services at little or no cost. And, Web browsers incorporate as standard features many attributes that are especially useful in analytic applications:

1. Local caching of pages (improves response time for repetitive analyses)
2. Viewing of partial results as page is loaded (good for speed-of-thought analysis)
3. Asynchronous processing, data compression, and data encryption (better and faster handli
4. Linking multiple data locations and data types allowing users to reuse data components

In short, Web access represents an extremely cost-effective way to provide widespread connection, achieving remarkable economies of scale, and simplifying IT departments maintenance loads.

4. Why Delay Creating a Web Front-End to Your Data Warehouse?

Anyone connected to the Internet is at least theoretically connected to everyone else. With the rapid growth of the Web, and especially the emergence of de facto standards, such as Netscape Navigator, Microsoft explorer, Java, and ActiveX; capabilities seem to abound. And we already mentioned that using a Web browser rather than client
front-end tools can save a bundle of money. What could possibly be the downside?

Security. *Until recently, the answer was easy and near universal -- lack of security.*

The thought of sensitive company information out there on the wild, unruly Internet neatly formatted for your competitor and an unsavory hacker used to be too scary to even think about. Encryption schemes and secure servers have allayed some of these concerns. HP's VirtualVault and Praesidium security solution set are examples of secure server components. Each of these components provide for single user signon and authentication, encryption capabilities, extensibility to public key (PKI) and Smart card access components, access across multiple platforms, and single point administration.

But the phenomenon that settled the "open to everyone Internet" issue is the intranet, at least for internal organization users. Security is maintained from the outside world by using network architecture components and software in a firewall complex. The firewall complex can screen inbound traffic through hardware component filtering and software in gate-way servers. Putting the Web applications behind a firewall makes them secure from the prying eyes of uninvited outside guests. Diagram 1 represents this Internet / Intranet implementation scheme:

Diagram 1: Firewall implementation scheme

A word to the wise still exists even with internal intranet users, however. Access to and sensitivity of the data being used on the corporate intranet still can be compromised.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Traditional client/server development basically controlled access to application data based upon who had the front-end software. If you had a right to know the information, you either had the software loaded on your desk top or had access to it.

The intranet web browser software is usually all that is required for the Web based application to be initiated. If an information consumer has connectivity to one Web application he potentially can access any of the applications in the Web application portfolio. Password restricting access to security sensitive information will be necessary to prevent unauthorized and potentially illegal access to Human Resources, Strategic Marketing, and Corporate Accounting data even for internal users.

Corporate Database dialog on the Web.

Another big unanswered question was exactly how to get the corporate databases to talk to Web applications. The CGI (Common Gateway Interface) protocols for message passing to and from the Web servers require writing code in procedural language like C, a step backwards in technology for many.

Luckily, Web application generation tools are starting to appear on the market. Application tool environments like NetDynamics, Allaire's ColdFusion, and Bluestone's Sapphire, among others, can generate both the application and the related Application Program Interface (API) calls. These toolsets hide some of the complexity of the low level interfaces from the development staff. The products are immature to the task at the moment, but the trend is in the correct direction.

If you are more inclined to programming rather than off-the-shelf applications, the advancements in Java class libraries, and/or Microsoft's ActiveX extensions, should make talking to data warehouses from the Web easier to code.

It is important to note that the tool approach for Web application creation can sometimes limit the hardware platform and database connectivity choices available to the developer. As noted earlier, many of the tools are in early stages of release to general availability. They tend to be more supportive of lower cost platforms like Microsoft NT and may support only a few of the available Data Base Management System (DBMS) connectivity Application Program Interface's (API's). Access to additional DBMS (API's) using UNIX and proprietary platforms have depended upon client demand. If your target platform, DBMS, or DBMS connectivity (API) is other than the primary development platform for the tool, vendor choices can be limited or beta software testing may have to be included in a project's roll-out plan.

This market is very aggressive and will expand to include evolving (API's) and DBMS as customer demand dictates.

Stateless Communication. *Interaction between the Web browser and Web server is stateless.*

Another problem in using the Web to access a data warehouse is that the interaction between the Web browser and the Web server is stateless. The familiar notion of connecting to a data source,

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

running the application to completion, and closing the connection does not exist in the Web world. The Web server acts as a message-passing server, responding to messages it receives. Without special coding, it does not remember the last action it performed or the last link that it used.

The message passing nature of the web server presents the possibility of crippling performance overhead. If each database query to support an application invoked through the Web browser requires the building of a connection and a database login, more resources may be spent managing connections versus doing actual work. The developer's choice, given the opportunity, is to let the DBMS vendors or the application generator vendors
build the Web gateways using CGI, or more recently NSAPI and ISAPI web server APIs. In later sections we will look at some techniques that can be employed to simulate sustained connection program calling states.

Work continues in the area of maintaining state information for Web applications at the server. Web browser and server product vendors are instituting the concept of "cookies", as seen in Netscape's implementation. The cookie is container where bits of information which are passed between each component to maintain an active connection. Examples of "cookie" information might be the user id for a request, the count of the last record passed for an SQL cursor to return the next batch of data, or parameters for the next SQL call transaction to complete in the application.

New standards are emerging to support the many development needs for applications living in a stateless environment. These needs include maintaining database connections and providing secure frameworks including encryption. Two such standards include Internet Inter-ORB Protocol (IIOP) and Standard Electronic Transaction (SET).

IIOP builds upon the messaging capabilities developed in CORBA-compliant object technology. It offers the possibility for an application to set up a connection through currently supported protocols, with the object being passed information to manage the connection and terminate the link when all work is completed. The object would be responsible to maintain the connection. Further dialogs would be conducted directly between the object and the caller. When the action of the object is complete, the object would close the connection. To operate IIOP, additional software layers will be required on the caller (browser) beyond what is on most desk-tops today.

SET builds upon the capabilities of transaction monitors to ensure secure transaction processing for electronic commerce over the Web. Similar to the IIOP dialog, SET would employ additional software on the calling browser to maintain the connection and perform security hand-shaking. SET employs an encryption method with the connection which is controlled by a unique security key, per connection. If the connection should be lost during the SET dialog all transactions would be rolled back to maintain the data in a consistent "before update" state.

5. The Data Warehouse / Web Architecture Approaches

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Now that we have looked at the product requirements being faced by Web technology tool providers, let's consider some options in architecting a data warehouse/Web solution and approaches vendors have taken. The solution to an organization's Web data distribution needs can be addressed with four deployment approaches. They include:

1. HTML Published Fixed reports
2. Linked web pages simulating drill-down
3. Application with fixed formats and flexible requesting mechanisms
4. Active Ad-hoc access

In addressing some of an organization's information delivery needs, just making reports constantly available for wider distribution may be all that is required. In such cases it has been found that Web distribution of data warehouse information does not need the warehouse as an active link processing queries. The data satisfying the majority of the users would be publishing information as a set of viewable reports. Options one (1) and two (2) address this architectural approach.

In the HTML Published Fixed report and Linked web page design approaches, reports are created in batch processes in the background and deposited to a report server. The data in the report server is stored in HTML formatted data pages which can be requested by the Web browser information consumer. The data is accessed from a report browser application which reads each requested report and formats the report into HTML tables for presentation. In the case of the linked web page approach the user is given the perception that he or she is drilling up or down to different grains of the data, when in reality they are linking to yet another stored report.

The stored report approaches provide for very good performance in that long running query results are stored in advance. They also remove the impact of multiple requesters asking for the same report data from the warehouse data server. Over the course of the day repeated requests for the same reports, some running several minutes, would degrade overall warehouse performance. It is also a much easier solution to construct technically because simple table constructs are built and managed by the report access software.

The stored report option, while presenting the data in a consistent format, does not provide for dynamic querying of the report data. Some tools provide a limited query capability allowing users to subset the information in the report or aggregate sections of the report. To provide the user more dynamic query capabilities requires application logic to be built.

The next two options, Applications with fixed formats and flexible requesting mechanisms, and Active Ad-hoc access provide active links to the data warehouse. These options provide the most amount of flexibility for the information consumer while presenting wide degrees of variability in performance and responsiveness to each request being made to the warehouse.

The application with fixed formats and flexible requesting mechanisms approach focuses on custom developed applications using Web based programming interfaces and toolsets. Prompting methods are developed to gain user selection criteria, SQL requests are formulated from some base columns to be presented and the variable selection criteria, producing a result.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Careful planning is required with active database links to the data warehouse. Developers need to ensure that indexing within the warehouse database is utilized and necessary thresholds are established to prevent large result set volumes. There are large opportunities for long run times to over-run the performance characteristics of the warehouse data servers. With careful planning to understand the workload, which may be executing along side the application in the warehouse, the fixed report application option can be a viable solution.

The active Ad-hoc access approach is the least implemented approach for Web access today primarily due to its unpredictable negative performance characteristics. The volume and width (number of columns) of results are uncontrolled. This lack of control pushes limits in formatting results and produces run times which may be longer than any browser user would be willing to wait. Ad-hoc access is still relegated to the client-server approach with analytical software running on the user's workstation.

The focus for the remainder of this paper will be on active links to the warehouse and the specific components which can be employed.

6. The Data Warehouse / Web Architecture components

Data warehouses for the most-part employ relational database management systems (RDBMSs) that use SQL to retrieve rows and columns of alphanumeric data. We will also make the general statement, that unstructured content is managed as HTML documents. The challenge in putting the data warehouse on the Intranet is in properly enabling SQL database access from an HTML browser, and handling the associated data buffering.

The general model for a data warehouse/web architecture involves at least three tiers. These include the Web Browser, Web Server, and Data Warehouse Database.

Diagram 2. Generalized Architecture for Web Access to a Data Warehouse

The Web Server is responsible for processing the messages sent to it from the many Browsers on the Intranet, and converting formats as needed. This communication is done with the Browser in HTML or Java, today. The Web Server also will host or communicate with another server hosting the Common Gateway Interface (CGI) applications or Java applets that are to be run.

The Web server application translates messages received and builds data warehouse SQL statements "on the fly". The queries may use pre-built query templates and supply final parameters to a database (API) . The database access (API's) fall into either proprietary RDBMS (API's) from relational database vendors (e.g. Oracle's SQL*Net or Sybase' DBLIB) or ODBC. The Web server application must process
the result set from the SQL query to the data warehouse and reconvert the data to HTML. A common way of presenting the data is to place the data into pre-built HTML tables (2-dimensional), and deliver those tables to the browser.

Data warehouse environments often require management components to allow administration and support staff to understand access trends and to accommodate change in meeting evolving business environment needs. The continuous addition of new fields, fields changing locations in tables, new

detail tables, and inclusion of summary tables will create challenges for users, and more specifically Web application developers, in interacting with the warehouse. Hewlett Packard markets the products, Intelligent Warehouse and DataMart Manager, to provide for performance monitoring, change management and security administration without affecting individual users and applications.

Intelligent Warehouse (IW) and DataMart Manager will preprocess queries on the behalf of the data warehouse database to decide the best way to satisfy the users information request. Since IW connectivity is established using ODBC, the dialog remains the same whether talking to a conventional client tool, or a Web Server. As queries are presented to the data warehouse through IW, the software intercepts the SQL query text and performs column and table name substitutions, summary table replacement, and other query optimizations. The resulting query is passed on to the Data Warehouse database layer. IW may also be used to perform joins on queries that reference data found in multiple Data Warehouse databases.

There are several variations to the Web server to application program dialog model, depending on the underlying technology that is chosen. In following sections of this document we will specifically look at three models:

1. CGI-based model. The more traditional approach, and probably still the most common.
2. Microsoft model. Based upon ActiveX extensions from the MS Windows platforms.
3. Java-based model. Using newer technology centered around the JAVA programming language and Java applet delivery model.

7. But First a Word on Firewalls

Our discussions in this paper have focused primarily on data warehouse access from an Intranet, not an Internet. That is, the entire user community and data are behind a "corporate firewall", thereby minimizing many concerns about external security. But what is a firewall?

Firewalls are the accepted technology for protecting a company's internal network from the Internet. The firewall mediates on acceptable traffic between the two networks, and implements a company-specified security policy, which usually states: "everything which is not explicitly permitted is prohibited." The firewall mediates all traffic using a filtering mechanism. The firewall: watches for suspect data passing through it's complex, and blocks all traffic that is not explicitly allowed by the organization's security policy.

The firewall is involved in all communications that cross the boundary between the internal and external networks, and provides protection over what "comes in" and "goes out" to the network. An example of a firewall activity is as follows: "A company's security policy only allows browsing and ftp (file transfer protocol) access for internal users to view external pages and transfer files from the Internet. The company's network administrators configured the firewall to allow these services and no others". Any network activity outside of these specific services would be denied, as in executing an external program or running a remote shell to an external system. Since no external user privileges have been configured, no external access would be allowed for any purpose.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Diagram 3: Firewall positioning

A firewall can restrict access to certain services and provide some screening of data that passes between users and a service, yet, it does not enhance the security of the platform on which the service runs or of the application implementing the service. If, for example, you install a "buggy" web server that mistakenly allows an external user to start a program on the web server machine (and that machine is attached to the internal network), an intruder could exploit that bug to attack the company's internal network. The firewall can restrict access to the web server machine, but does not enhance security on the server machine.

8. How Does CGI Work?

To allow an application to dynamically generate a set of results from a user request, the Common Gateway Interface (CGI) has been provided. CGI is a standard for interfacing external applications with information servers, such as HTTP or Web servers. It essentially allows for the launching of a specific application which processes a transaction with the expressed purpose to produce a result set and optionally build a new formatted web page. The CGI application may be passed parameters from a form in the web page displayed to the user. This process flow is represented in diagram 4.

Diagram 4: Basic CGI Process Flow

The basic data warehouse connection to the CGI application is represented in figure 5. Note that a web server initiates a dialog with the CGI Formatter which acts as the front end to the data warehouse. In most application deployment schemes the CGI application and the warehouse are not likely to be on the same hardware box. This
will require the selection of a DBMS (API) to execute queries against the database. Most commonly, the Native DBMS (API) is selected , but ODBC can be chosen as well.

Diagram 5: CGI connection to the Data Warehouse

CGI applications must be concerned with not only security and translation of the request into appropriate SQL, but they must then create an HTML page based on the results. Product suppliers have moved in producing products to provide a forms/reporting layer, referred to here as a CGI Formatter. The CGI Formatter removes much of the formatting logic from the application, and may even remove the need for a custom CGI application altogether.

The CGI formatters now hitting the web market can receive and process requests without the creation of custom application code for data requests to the data warehouse database. Example formatters are: NetDynamics, Speedware, Cold Fusion, and others.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Performance issues do occur with the model represented in diagram 5. Typically, each client transaction requires specific steps before an SQL transaction can be issued to the DBMS. These are:

1. Load the CGI application into memory
2. Start up the CGI application
3. Open the connection to the database

Each CGI transaction is treated as a single program execution. For applications requiring several CGI transactions to complete the information consumer's data request, maintaining state information requires passing information forward from the CGI application back to the browser so that it can be appended to the next transaction . This makes maintaining the scope of a logical transaction from each CGI physical transaction difficult and limited.

To overcome these limitations, the application logic may be moved to an application server process which maintains state information (including database connects) between CGI calls. A light-weight or "thin" CGI application is created which can be launched rapidly and can pass requests to the appropriate application server. This is illustrated below in diagram 6.

Diagram 6: CGI Application Flow

This process can work for any CGI application. The formatter may actually run on another server, in which case the lightweight CGI application would perform the coding of routing information for delivery of results to the requester.

Introduction of application development tools and 4GL's adds an additional computation and conversational slant to the CGI application. An example of this application process flow is shown in diagram 7. Here a session is established and dialogue commences between an application and client using HTML/HTTP as the protocol.

Diagram 7. Application tool and 4GL processing flow

The initial request starts up a session for the application which responds with an HTML page. The HTML page contains the first dialog and embedded CGI calls with variables necessary to establish the application context. If a light weight CGI application is used, one or more of the variables indicate which instance of the full application should be used.

9. Advantages and Limitations using CGI

To summarize CGI application design, CGI provides a modular approach to building Web applications. Based upon its message passing design, CGI allows application logic to be placed on any server. The CGI module can be reused by many applications and linked to combine various data from databases, text servers, and multimedia for an end user's consumption. The applications are open ended using API calls to dialog with databases. This allows single points of access to be built to the database which can be altered as choice of database or access protocols are changed.

Despite the advantages of CGI, at times it would be more efficient to interface and run queries directly within the Web server itself. The CGI interface requires the server to set up and execute a program each time it is requested. This startup operation is very expensive to the server's resources. If a CGI program is not run very often, this is not a problem. But there are many situations in which a CGI program must be frequently accessed. For these situations it would be helpful to have an integrated solution.

As previously mentioned, the basic issue with the CGI interface presented to the data warehouse is that it does not offer a continuous connection to the database. As a result, it is impossible to support an application requiring multiple interactive queries -- a data warehousing requirement. One approach to solving this problem is to employ a message-based protocol between the client browser and the server-resident analytic layer using CGI. This layer would be responsible for mapping a user to a server account and starting a process that executes as that user. A continuous connection would be set up and maintained between the logic layer and database. Processing of the iterative queries could be guaranteed over the lifetime of that process with minimal system resource consumption for the connection.

Let us look at this in finer detail. For example, an HTML form can request the user to enter his/her UNIX user name and password. The password is encrypted before being transmitted over the network. The information is then passed as parameters (or through environmental variables) to a CGI program, written in C, C++, or as a Perl script. After de-encryption, the CGI program verifies the UNIX user name and password via operating system security features and starts a process that executes as the user. This "surrogate" process can then connect to the RDBMS using a DBMS API (such as Oracle's SQL*Net or Sybase's DBLIB) or to the Intelligent Warehouse middleware (which will pass the username/password through to the RDBMS), and sends the user name and password. Once the connection to the RDBMS is established, many queries and result sets can be processed. Data buffering is performed in the database server (and in the IW layer if it is present). A final output report can be generated, converted to an HTML document on the Web Server, and sent back to the client browser for display to the user.

The CGI interface in its basic form is unable to share data and communications resources. When a CGI program is accessed by a client, a new copy of the CGI program is invoked for each remote client. If your program must access an external resource (such as an IPC pipe to another resource), it must continually close and re-open that resource. Depending on the demand for the resource, this opening and closing can have worse effects than the start-up outlined earlier.

Another limitation of the CGI interface is that it is designed for a very specific purpose: returning the data to a network navigator. This in itself is a very powerful feature, but there are times when it would be nice to have your own custom functions perform some aspect or even aspects of the request-response process.

10. Web Server (APIs)

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Some of the limitations of various Common Gateway Interface implementations have given way to the development of Application Program Interfaces (API's) . In an attempt to improve web server connectivity performance, both Netscape and Microsoft have developed their own specific (APIs), known as NSAPI (Netscape's) and ISAPI (Microsoft's).

Understanding that the web server is the point of control for the application, these companies have given their web servers additional functionally much in the same way as control programs of the past; through user exit logic. User exits have been part of control programs like IBM's CICS, operating system sub-routines, and software distributors applications for a very long time. Following is a brief description of these new web (APIs); please refer to the respective vendor's Web pages for more detailed implementation information.

NSAPI (Netscape Server API):

Netscape provides an API for adding functions to the Netscape Web Server. According to Netscape [10],

> 'The NSAPI was designed to solve performance and efficiency problems common to installations that make liberal use of CGI functionality.'

One can override server functionality or add to it by developing functions and then dynamically linking them to the Netscape Web Server. The developer does not have access to the Netscape code, but can access key variables. The interface consists of a set of environment variables that the server uses to send information to the invoked program. The nature of functions being linked dynamically is key. The dynamic linking allows functionality to be altered over time without taking the server off-line to re-link its component parts.

ISAPI (Internet Server API):

ISAPI is the server extension API supported by Microsoft's Internet Information Server. Instead of using an executable module, ISAPI uses Dynamic Link Libraries (DLLs). These DLLs are loaded into the memory space of the server; the application runs in the address space of the server. This technique increases performance by keeping the code cached in memory instead of having to reload it for each request. ISAPI allows interactive response to HTTP requests, instead of simply returning HTML files. It is run in conjunction with a Windows environment, and locks you into using the Internet Information Server (IIS).

Microsoft quoted the following advantages of ISAPI over CGIs are: [9]

Speed -- There is a considerable gain in performance.
Features - ISAPI allows for creation of server filters (for pre/post application requ

In general the API approach does provide for

1. Improvements in run-time efficiency
2. Ability to extend the functionality of the Web Server component

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

3. Provides a consistent way for the application development staff to add functionality to the server.
4. Allows the Server vendor the ability to integrate DBMS functionality much tighter than the CGI approach.

The disadvantages the of API approach compared to CGIs:

1. Forced standardization to the web server of choice
2. Only a few servers currently support the API's.
3. Ease of Development -- Documentation is very sparse, and debugging is somewhat tedious.
4. Potential inability for two servers to communicate

11. A CGI-Based, or Web-API based Architecture

Now that we have explored the individual components of a CGI-based OpenWarehouse Web solution, let's put them together in the same framework, as shown in the next diagram

Diagram 8: Web CGI and API framework

Here is the review of the architectural components we have now discussed:

CLIENT:
The HTML browser is the software used by the end user to gain access to the web based application, normally either Netscape (65% of market), Spyglass, NCSA Mosaic, or Microsoft's Internet Explorer (20% of market, but growing)

HTML Forms and Templates are the components that support the HTML display on the PC. This is a generalization of a number of functions available in HTML for forms-based, images, etc.

INTERNET / INTRANET CONNECTIVITY:
The Internet Connectivity is a "cloud" of mostly networking protocols (URLs in HTML) by which a client-side request for information from a new URL is matched to the correct server for the resource. http (short for HyperText Transfer Protocol) is the typical manner for interpreting the content, and TCP/IP is the generally accepted network protocol.

Part of the Internet / Intranet connectivity is the inclusion of the corporate network's firewall complex. As mentioned earlier, the firewall can and will filter dialog both inbound and outbound to the cloud. Additionally requests can be sent on behalf of a user from another web server. This second web server acts as a "proxy server" to the user. The firewall complex, not pictured here but residing in the http Internet connectivity "cloud", can have filtering mechanisms as well as security components.

WEB SERVER:
The Web Server is responsible for processing the messages sent to it from the many Browsers on the Intranet, and converting formats as needed. From here, the CGI programs can be launched using either a general CGI interface, or the browser-specific APIs (NSAPI or ISAPI). Also here, requests to the

database server are formatted as SQL text, and sent either 1) through ODBC; or 2) through RDBMS-specific (APIs), to retrieve the requested information. That information is reformatted in the Web server, and presented to the client's browser as HTML text.

INTELLIGENT WAREHOUSE and DATAMART MANAGER
Intelligent Warehouse and DataMart Manager are Hewlett Packard's data warehousing database middleware products for handling the following difficult problems with large data warehouses:

Summarization navigation (mapping the user's query to the smallest summary table that is sufficient to provide the correct answer in the shortest amount of time).
Virtualization of the user's schema. Allowing the business community to build their view of the data warehouse as defined by the business needs, while the operations people can use more precise physical names for columns and tables.
Query Blocking. Canceling those queries that might provide an inconsistent answer due to incomplete data in the database.
Heterogeneous Database Environments. Joining data from two or more databases, perhaps from two or more different RDBMS providers.
Query performance Monitoring. IW logs all queries it performs, and can be used later by a DBA to analyze query traffic, table usage, and general query performance.
. Value Based security: Automatic inclusion of "where" clauses to restrict access.

DATA WAREHOUSE:
Your data warehouse is a database, specially built for high-speed query processing. It can be built upon general RDBMS' such as Oracle, Informix, Sybase, Red Brick, or others.

OTHER DATA SERVERS:
The data warehouse is a only a component in a Web-enabled application. The data warehouse could have URL'S stored in its data content that the Web server could link to retrieve additional information for the user.

An example of this might be where product information is stored in the data warehouse with a column used to specify the location of the product's literature. . When chosen, the URL location of the product's literature could be linked to by the web server presenting the information retrieved from the data warehouse along with the literature sheet. Many other examples exist in Real Estate, and Product Marketing.

12. Java-Based Architectures

Beside the usage of the CGI and (API) to web server approaches, there continues to be a ground-swell of investigation, testing and support for the language specification called JAVA. It represents yet another approach to placing program functionality both the web server and at the client requesting the service. Let's explore Java in more detail.

12.1 What is Java?

Java is a new 3GL application language / environment, though not initially developed for the Web, has been extended to provide Web specific features. It is analogous to a subset of C++, but with features that overcome C language based development issues. These include:

* Replacement of pointers with arrays and runtime bounds checking
* Easier, more reliable memory management
* Interpretative, platform independent code generation by compiler
* Strings are inmutable,

To these were added:

* Notion of applets
* Security

The key to Java, with respect to the Web, is the provision for transparently downloadable applets. The applets are small applications stored on the server and which are downloaded to the client for execution in the web browser's environment as needed. By using an applet, one can build a more dynamic interface instead of the relatively static presentations of HTML pages. The applets could additionally take advantage of special functionality stored on the local client desktop to enhance application performance.

Since applets are programs which may come from an untrusted host, there are special security constraints imposed upon the applett when executing on the client desktop. These are:

1. Applets may not read, write or perform directory operations on the client's file system.
2. Applets may not make network connections except to the host from which they were loaded (originating host), or a host specified alternate server.
3. Applets may not have native (compiled, non-interpretive) code.
4. Applets may not fork (branch) or start other processes. Although they may be multi-threaded, they may not manipulate other threads.
5. Since Java does not permit pointers, a Java program cannot manipulate memory outside of their own data areas.

Diagram 9: Sample Java Dialog

These constraints are imposed by the client browser, which is responsible for interpreting the Java application code. The above are correct for Netscape's browser. Sun's browser allows the identification of special applets which may be violate these rules. The upcoming versions of Netscape is are also expected to allow modification or tuning of the security constraints.

Applets may also be loaded locally from a file system instead of over the web. Local applets loaded through the file manager are **not** subject to the above security constraints.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Lastly, with respect to security, Java is a published standard. The source code is public to invite independent audits and challenges. It does not depend on hidden techniques to provide security. As a consequence we expect Java security to become very robust.

12.2 Java-Based Web/Warehouse Architecture

The architecture for enabling Java-based communication to your data warehouse is not much different from the architecture we explored for a CGI-based architecture.
Diagram 10: Java Architecture

A Java Virtual Machine (JVM) component exists on the client machine. It is a set of files and processes consisting of common Java class libraries. The Java Applet is downloaded onto the client (and into the JVM) as it is requested by the user. The master copy of the applet is kept on the Web server, and launched from the Java Application, which also runs within the Web server.

The database connectivity layer can either call a native DBMS API mechanism directly or exercise an interface to a specific service. One such service is JDBC. The JDBC interface performs any necessary Java to ODBC protocol conversions and related message traffic to/from the data warehouse. JDBC is an evolving interface and shows much promise in improving upon both the functionality and performance of ODBC.

12.4 What Are the Issues with Java?

Among the areas in which Java could use some work are the following: [1]

Bandwidth. Anyone who's encountered a Web page with even a modest applet knows how long it can take for the applet to download. Remember the applet needs to be downloaded before it can execute. With many users all requesting service from the same web server, bandwidth demands affect the web server and the networks that they are attached to. The problem is especially acute when a server has to serve up customized HTML pages upon demand.

Performance. Java applets not only take a long time to download, they also run more slowly than programs written in other languages. The Java appplet is interpreted code so that it is able to execute on any platform unchanged. The applet suffers from not being a natively compiled executable, able to take advantage of the system it is running on. Because of the interpretive way which JAVA uses CPU resources, Java is estimated to be up to 50 times slower than C++ at performing certain tasks. New just in time (JIT) compilers are emerging which will compile the applet when downloaded to address the processing speed issue.

User acceptance. Tools such as Macromedia's AppletAce and Aimtech Corp.'s Jamba are bringing Java functionality to the non-programming Web developer, but these remain tentative first steps. Java is a
complex language on the magnitude of C++ and demands a great deal of skill and knowledge to implement.

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION

Security. "The nature of Java security means that applets will have to remain severely constrained if we're to have any [security] confidence at all", says Silicon Graphics developer Hank Shiffman [1]. Most of the things that people want to do are not possible in an automatic download environment."

Setting the hype aside, Java appears to be a viable technology for the foreseeable future. Anticipated upgrades to Java's performance and security features will make it even more valuable. For high-performance data warehouse web environments, CGI may continue to be the correct Intranet connectivity solution.

13. Client Side Database (API's)

Several products are coming to market which provide RDBMS (API's) on the client side. JDBC is one of these. The idea behind these client side (API's) is to attempt to remove some of the many layers of application logic required to get database information in a Web environment. Reviewing diagrams in this paper has hopefully shown that when a user can request information from any where or on the behalf of someone. This means that logic must exist somewhere to do the navigation.

CGI applications maintain the database connectivity on the server side, as opposed to creating DBMS transactions on the client. A CGI application can become a gatekeeper for database requests, by providing blocking and queuing mechanisms for resource requests. As we address more of the Wide Area network (WAN) space, this gatekeeper position can present a bottle neck for user access.

To address the possible ramifications of a CGI application preventing users to contact the database, a client side (API) can remove the middleman and potential bottlenecks. The applications using such an (API) have to be well behaved in that less of the request mechanism will now be examined . The database server can be flooded by clients 1) performing arbitrary queries, 2) waiting long periods between commits, 3) receiving malicious attacks. Consequently, unchecked, client side database (APIs) run a greater risk of consuming database resources.

Furthermore, if a general (API) is available, one is potentially giving up control of the application generating the SQL. This is opening up the server to arbitrary SQL increasing the risk huge consumption of server resources and loss of security.

How are these different than the standard client server scenario? In a simple case there may be no difference. However, differences can appear due to: 1) internet access outside of the safe haven behind the firewall, 2) the general model of web / internet usage is that clients may be more removed from the centralized resource, moving
further along the paradigm defined by enterprise warehouses, and 3) the client apps for JDBC / Web browsers have not yet been built. We may find other differences.

Finally, we need to consider one of the major motivators of the web expansion. That is the value of the widely distributed, thin client to a centralized resource. Will client side database (API's) work well in this model or will they lead to fat clients with difficult distribution and support mechanisms?

14. SUMMARY

It is clear that the Web is here to stay, and can play a very valuable role when connected to a data warehouse, even in the short term. Selecting the correct components, and building a flexible architecture that can withstand exponential usage growth, is the exciting part in engineering an Intranet/data warehouse solution.

BIBLIOGRAPHY

[1] Pomeroy, Brian, "Jammin' or Jivin': Is Java the Internet's Breakthrough Technology?", High tech Careers, Nov, 1996, pp. 8-12.

[2] Raden, Neil, "Warehouses and the Web", InformationWeek, 13 May, 1996, pp. 80-86.

[3] Tanler, Richard, "Putting the Data Warehouse on the Intranet", Internet Systems, May, 1996, pp. 34 -37.

[4] Watterson, Karen, "GUIs: The Window to the Enterprise", Data Management Review, Oct, 1996, pp. 67, 71 .

[5] anonymous, "Challenges in Managing and Using a Data Warehouse", HP White Paper, Oct., 1996.

WEB PAGES

[6] http://www.hp.com/go/datawarehouse -- general HP OpenWarehouse info
[7] http://www.dmo.hp.com/gsy/solutions/dw/datamart.html -- HP's DataMart Manager product announcment

[8] http://www.data-warehouse.com/articles -- DW-related articles
[9] http://www.iftech.com/oltc/isapi/isapi1.stm -- "Developing ISAPI Extensions" article
[10] http://home.netscape.com/newsref/std/server_api.html -- Netscape page on NSAPI
[11] http://pwp.starnetinc.com/larryg/whitepap.html -- DW white papers collected by industry consultant
[12] http://pwp.starnetinc.com/larryg/index.html -- Index to information on DataWarehousing (from Larry Greenfield)
[13]http://hoohoo.ncsa.uiuc.edu:80/cgi/intro.html -- CGI background information
[14]http://home.netscape.com/home/handbook/docs/learn.html -- Netscape's Internet primer

ARCHITECTING A DATA WAREHOUSE/INTRANET SOLUTION