DB2 magazine

http://www.db2mag.com/db_area/archives/1997/q3/97fabeu.shtml

A Hyperlink to the Mainframe

Hooking your legacy systems to the web doesn't have to be a tumultuous experience. Here's how one company succeeded through resourcefulness and trial and error.

By David Beulke

Fall 1997

Everyone is surfing and discovering the Internet's tremendous capability to distribute and provide access to information. Since thousands of Internet startup companies are being funded by venture capitalists, I thought I might be able to get funding by getting experience using this new software and hardware environment at my current company.

I figured it never hurts to have experience with well-funded technology, so I decided to put together a pilot project displaying mainframe DB2 data on the Internet. This article details what I learned while putting the pilot together. When embarking on an Internet project, the first thing you discover is that everyone wants to help you, but no one knows where to start. I found that the best place to begin is on the Internet itself. The amount of information that I found on database connectivity was mind-boggling. All I wanted was to take our existing database systems and put their content on the Web, but there are a number of decisions that make the task quite complicated.

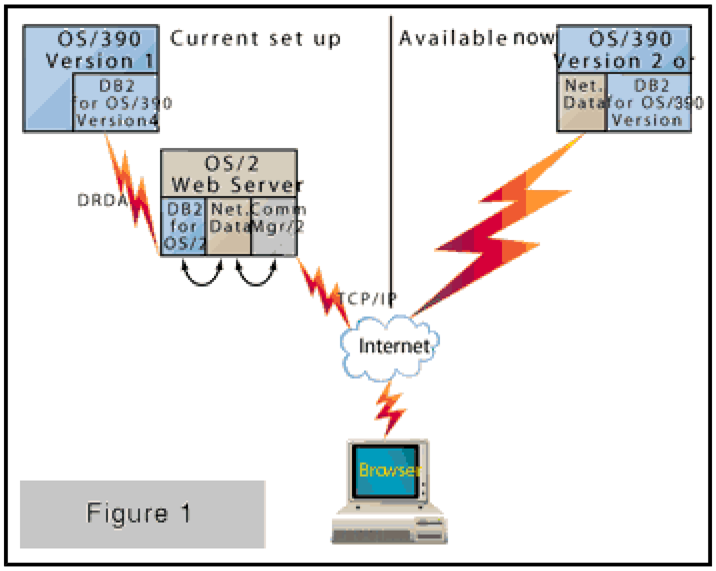
CHOOSING A SERVER

The first decision to make when setting up any Internet solution is which server you will use to serve your data to the Web. The good news is that with the release of OS/390 version 2, the mainframe is becoming the biggest and most robust server available. Version 2 has native Internet protocols (TCP/IP) and a Unix kernel. (A Windows NT operating system kernel will be added in version 3 to run anything from any platform.) The bad news is that my shop doesn't have these technologies installed yet.

My solution for the pilot project was to use an OS/2 server with DB2 for OS/2 and a Communication Manager/2 connection to access the mainframe DB2 information. The OS/2 connectivity configuration could be substituted for any platform or system that can currently reference mainframe DB2 systems: You can use Windows platforms with ODBC connections, Sybase connectivity, or any of a number of vendor solutions as the gateway to your mainframe data.

If you have the version 2 release of OS/390 installed, you can hook up the mainframe through TCP/IP or as a Windows/NT server directly, thus eliminating any intermediary platforms (see Figure 1). Even if you can't get your people to upgrade to the new OS/390, you can get direct

TCP/IP connectivity through DB2 version 5. Version 5 also provides the option of storing data in ASCII format - the same format found in your PC and the Unix environment - making it easy to share data across the whole enterprise through the mainframe. The v.5 ASCII storage capability is even more important in an Internet context because the Internet needs the data in ASCII format.



OS390 v.2 or v.3 and DB2 v.5 provide an easier solution.

Since I had only an OS/2 server configured for my project, it had to translate the data from the mainframe EBCDIC into ASCII format. The small OS/2 server could become a bottleneck if a large amount of data needed to flow through it. Because the translation process is CPU intensive, make sure you have enough CPU horsepower if you have a small gateway translation server. Of course, if you have the ultimate configuration - OS/390 v.2 and DB2 v.5 - you will have direct TCP/IP connectivity and data stored in the ASCII format for the best performing Internet server solution.

With help from some network buddies at my company, I began to configure our intermediate OS/2 server. We ran into a small, but very significant, problem: The communications protocol's code-translation pages specified for the server and the mainframe were incorrect. This translation problem was caused by a typo during the setup of the OS/2 server communications protocols. This meant that the translation of our DECIMAL, INT, and SMALLINT binary fields from the mainframe EBCDIC into the ASCII format for the Web browser were incorrect. Going back into the communications setup, we aligned the server's communication code pages, and our fields translated correctly.

WINDOWS CONNECTIVITY

Another difficulty was establishing compatibility between the various Windows connectivity products on the project's gateway OS/2 server. The OS/2 server was hooked into and accessed via a number of different configurations, such as an MDI gateway to access CICS, an ODBC configuration for referencing server data, and a DRDA configuration to access mainframe database systems. Each configuration has its own set of issues and complications, and, with all of them, we ran into compatibility issues between the various Windows gateway product releases. The Windows software was not aware of other versions or other connectivity software on the server. These Windows releases wanted to share the same libraries/directories, so we had to deinstall them and install the one we wanted to use for connectivity to the mainframe. Because OS/2 can talk to just about anything from Windows NT to the office humidifier, you need a good understanding of the software and its release compatibility before attempting this kind of connectivity.

NET.DATA

The next thing our OS/2 server needed was to be able to pass SQL from the Web browser, through the server, to the mainframe. This process is supported by a new piece of software called Net.Data, which can be downloaded from IBM's Web site (www.software.ibm.com). Net.Data allows Web browsers to access and communicate through our OS/2 gateway environment. (IBM's Web site also contains examples of HTML pages with DB2 SQL and additional setup information. These examples and downloadable software are great for getting a skeleton system working quickly.)

Net.Data is a small but powerful piece of code used to put into a gateway environment; its application program interface (API) code is bound against the server's DB2 instance. Since Net.Data is bound into the gateway environment, we used normal DB2 security to restrict access to the mainframe. This API is then used by the Web browser HTML scripts when they specify the SQL_EXEC command to pass the SQL and arguments appropriately.

Testing this Net.Data configuration was simple: Once we thought we had it set up correctly, we set up a Web browser on the server itself, then wrote a simple HTML page designed to only put a SQL statement into the system. The SQL statement we used was the one we'd downloaded from IBM's Web site. We then looked at our mainframe DB2 system's monitor and verified that the SQL made it through to the mainframe. Luckily (after the fifth try) we were able to determine the server security, network communications, and DB2 mainframe authorizations to make it all work. (An additional consideration we had with our Net.Data OS/2 server configuration was that all the SQL passed through it to the mainframe is executed as dynamic SQL, which needs to be prepared within DB2 and security checked against the user ID and various tables before being executed. For online reponse time this can be a large part of the processing, but for our tiny pilot project we didn't really worry about it.)

Once we had verified that our OS/2 Web server was set up, we noted its TCP/ IP address - the address people will use to access the mainframe information through the server. Based on your configuration settings, this address may stay the same, or it may be assigned dynamically. The scope of our pilot project's Internet connectivity was determined by its overall accessibility through our company's network. If our server could be referenced outside of our company's local network, the security on the server and the mainframe could be viewed by anyone who knew the server TCP/IP address, which would have exposed it to hackers.

OLD ISSUES, NEW TWISTS

The purpose of our pilot project was to retrieve and publish existing legacy database systems information. Leveraging our current data was important not only because we could use the existing infrastructure for other business functions, but also because the size of our databases and issues of data timeliness, data quality, and data replication made our legacy systems the best option. We also hoped that leveraging our current business data would help us avoid the system outages, disaster recovery, job scheduling, and operational support issues so inherent and troublesome for non-mainframe platforms and systems.

When we began to design our Internet system, we ran into all the classic software-development issues, but with new twists. Security was the first major issue and the most important factor in deciding what information we should make available - in other words, what information could be used or misused. We knew all the stories about security breaches on the Internet. Displaying credit card information is obviously a no-no. Inventory levels or customer address information could be used by competitors or for fraud, so that was also off limits. The discussions of what data to make available turned out to be so extensive that - just for the proof-of-concept project - we shelved the issue and asked for a user ID and password through our first HTML Web page.

Development costs were the next issue. We found out early on that the Web was cheap virtual real estate. The costs of setting up our HTML Web pages were extremely low, and, since we were only trying to do simple text displays, the forms were not complicated. We were also able to copy sample HTML from other Web sites. (We just had to make sure that we didn't borrow anything copyrighted or trademarked.) This cheap virtual real estate also highlighted our lack of planning; we were as guilty as sin because we didn't have a speck of documentation of where we were putting source Web pages, connectivity software, or paths. The configuration was working, but we got into trouble when someone went on vacation. It's essential to document the different mainframe network logical unit addresses (VTAM LU names) that the Communication Manager uses to connect the server to the network and to document which connection and database must be active for your Net.Data connectivity to work.

HTML editing and Web navigation issues were also new to us. We had all written scripts before, but trying to get HTML entry variables and pages to work was a lesson in finding the right example and copying it. Again, the examples we downloaded and the simple Web pages we copied were critical to our success. The HTML to display text data on a Web browser is very much like a script file with variables, areas for the data, and action buttons that can be mouse-clicked. It's easy to put these three simple HTML areas together, accept input data from them, and invoke different pages based on that input. However, Web navigation features can be difficult to design because of the sophistication of the typical Web browser with its ability to point and click on any portion of the GUI screen or any displayed object. In comparison to a traditional mainframe CICS screen, Web GUI design requires knowledge of list boxes, scroll bars, links to other Web sites, navigating to other HTML, and animating screen objects. Web navigation can also be programmed through several different language scripts, such as Common Gateway Interface (CGI), Perl, ActiveX, and Java. These script languages also offer various other browser controls and capabilities.

We set up a first page for Web users to enter their IDs and passwords. After having passed the security check, a user is asked for key information to search for and retrieve the desired data. After retrieving data from the DB2 system, the user is able to browse through the mainframe information.

It took approximately four full days over the course of two weeks for our group (which consisted of a DB2 dweeb, a server hack head, and a network nerd, all lacking in HTML experience) to set up our demonstration pages. Although we started from scratch, we were able to get everything going and display our pages with the help of several books, examples, and some trial and error.

Our little proof-of-concept project got us the knowledge we needed about hardware and software technology for hooking existing mainframe information up to the Internet. The ability to distribute a message to millions of Internet users is a powerful one, which is why venture capitalists are funding so many Internet companies. Now all I need to do is explain all my knowledge and Internet savvy to a venture capitalist. Know anyone?

David Beulke has been building database systems for over 13 years and is currently the Director of Corporate Data Management at Bell Atlantic, where he directs and manages all aspects of the enterprise's DB2 family and Sybase database systems. He is on the IDUG Board of Directors and was in the early release program for DB2 version 5. You can reach him by email at dbeulke@compuserve.com.