

# Implementing IP-based Object Methods in a Real-Time Web/ORDB Server

*Direct web browser access to distributed on-line database information is an important element in providing high Quality of Service (QoS) for real-time web sites. A uniquely powerful capability allows Polyhedra to function as a web server allowing web browsers direct access to data running in main memory. As a web/database server, it can deliver pages in real-time using information updated from a variety of external data sources. The ability allows high-speed on-line read/write operations to significantly more concurrent web browsers. Another unique feature, Active Queries, provides a fine-grained SQL Push Technology that is built into the database.*

## INTRODUCTION

**E**fficient Web technology-based real-time solutions can be built using the object-oriented features of a commercial-off-the-shelf (COTS) real-time database. Polyhedra is a real-time main memory (the industry's first commercial MMDB) Object-relational database that includes TCP/IP base classes and a control language for defining object methods to control behavior. Objects can be encapsulated with specialized methods to support any IP-based protocols including, HTTP, SMTP, NNTP, and SNMP. HTTP methods allow web browsers to connect, read, and write directly with database objects and processes running in main memory. Web browsers communicating directly with objects running in the database process itself means the database is the Web Server and the Database Server. The active nature of the database provides a unique feature called Active Queries that provides a built-in SQL push technology over IP.

## TYPICAL WEB SERVER TECHNOLOGY

A web browser is a program that can gather and display information from 'web servers'. To do this, it connects via TCP/IP to the server, issues a request (using the HTTP protocol), and interprets and displays the results. The 'files' delivered by the web server are typically textual, in HTML (or XML) format, or graphical, in which case they will usually be graphics in GIF, JPEG or BMP format. The HTML files contain hypertext markup instructions: not just the raw text to be displayed, but also instructions on how the text is to be laid out, the graphic images to display in the document, and the action to be performed when various links are to be followed.

Typical web servers get their data – the graphic or textual 'pages' they deliver to the web browsers – from individual files on their local disks. Designing a web server consists of setting up a server, making a selected directory accessible, and then populating that directory (and subdirectories with HTML files and images). Where dynamic pages of data are wanted, though, simple and direct file access does not suffice.

Databases with Active Server Pages (ASP) can be used as a dynamic page server mechanism. This typically requires several layers to process including ODBC for database access. In its simplest form, to get a web page where some or all of the contents are generated as the result of a SQL query, the web server runs a separate program which reads a template file, connects to the database then queries the database. Finally, it uses the template and the results of the query to dynamically generate the page(s) of information and returns it to the browser. A helper application – often referred to as a 'cgi' program – is started up each time one of the dynamic pages is requested by a browser, which obviously affects performance.

## REAL-TIME WEB/DATABASE SERVER

With the active Object-oriented nature of a Polyhedra database, new and more powerful approaches to building applications on real-time web database servers are possible. The simplest is to use the built-in class support for TCP/IP and define an object in the database that allows the whole database to act as a web server. Individual tables (classes) can contain more specific object methods.

To simplify the task of web database developers that want to make their database function as a real-time web server, they can use the examples that define two

classes: `httpServerBase`, which handles incoming connection attempts by browser clients, and `httpConnectionBase`, to handle individual connections. When the web/database server sees a web browser is trying to connect to the database process, an `httpServerBase` object creates an `httpConnectionBase` object, and it is the latter which reads and analyses the request, and generates and sends the response directly to the browser.

To be more accurate, the `httpConnectionBase` calls a virtual method called 'Respond' to work out what the response should be, and the results of this method are returned to the web browser. The database designer can derive his or her own "specialized" class from `httpConnectionBase` and redefine the Respond method.

For example, the Respond method could be redefined so that:

- the requests "GET /" and "GET /home.htm" iterate over a table in the database, and return an HTML page consisting of a table that lists the name and value fields for each record found in the database table;
- a request of the form "GET /<name>.htm" will locate the corresponding record in the database table of interest, retrieve historic information about earlier values of the record, and return the results in tabular form; and,
- any other request will result in an error message being returned.

Adapting a Polyhedra-based application to become web-visible requires only the definition of this function (whose complexity depends on the complexity of the views required) and the required CL required the SQL definitions.

How would one use such a system in a live application? There are a number of choices. Firstly, you could make the real-time web/database the main web server, by adding in classes for holding the static pages – assuming that there is enough memory for this. An alternative is to use a traditional file-based web server (such as Apache) to hold the static data (which in many applications will be the bulk of the pages), and have the real-time web/database server, (configured for fault tolerance) hold the dynamic data to support intensive query resolution and heavy transaction processing.

In order to protect the real-time web/database server from potentially-malicious browsers, it is possible to 'hide' it behind the other static web server, which would act as a filter/cache; users would not be aware that certain pages were being created by a separate web server.

A real-time system based on such an architecture delivers a significantly more efficient way to provide the target application with a GUI. A greater number of concurrent online browser clients can be supported on comparable real-time platforms than conventional systems with separate passive databases and web servers. Take a look Linux-based [www.polyhedra.com](http://www.polyhedra.com) as an example. Control Language (CL), Polyhedra's internal object-oriented method scripting language, has been used to implement the

HTTP 1.0, SMTP and NNTP protocols as object methods. The web server is a Polyhedra database that contains all of the information viewed online and uses no other web server technology. Pages are dynamically created on the fly. For Polyhedra's own Intranet access, the server offers tables of web access history and information request etc. Also, information requests are emailed automatically to company personnel with acknowledgements emailed back to the requestor.

## REAL-TIME FAULT TOLERANT ORDB

Polyhedra servers are viewed and accessed as "Relational Object Stores" with SQL access, manipulation and definition provided with SQL that has been extended to support inheritance, methods, and user-defined datatypes, among other things. In addition to snapshot data persistence, Polyhedra provides a sophisticated Historian for high-speed data journaling and storage to disk. The data stored via the Historian is accessible via static and active SQL.

To support continuous operation, fault tolerance is provided with an Active and redundant Standby database configuration. A standby database is kept up to date with the changes that occur in the active database, and is ready to go live as soon as it is told to do so – either as a result of a system crash or because a controlled switch-over was requested. The roles of the databases can be switched at any time and Polyhedra provides several configurations to decide which of the two databases should be Active and which should be Standby. The Standby database is on-line at all times and accessible to clients for read-only queries.

Using a standard TCP/IP connection, the Polyhedra generic device interface API allows data acquisition for updating the database tables in real-time.

Polyhedra is processor, operating system, and TCP/IP stack independent. It can be used as a web server/database on any supported operating platform including; VxWorks, LynxOS, Enea's OSE, and pSOS, most popular UNIX implementations, Linux, and Windows NT. ■

---

*Nigel Day, Ph.D. Since gaining his PhD from Cambridge University Nigel has been involved in a wide range of leading edge software design and consultancy projects. Experience includes compiler and operating system design and implementation, and computer security research. Nigel is currently Technical Director of Polyhedra Plc., serving on the Board of Directors, and as Vice President of Engineering for Polyhedra, Inc.*

*David C. Morse. Mr. Morse has been working in various marketing and management positions in the embedded database industry for the past 15 years. As the founder of database tools and technology company in 1991, his experience with a variety of database and RTOS environments has allowed him to produce articles and technical papers on various subjects in this industry. He is the founder of Polyhedra, Inc. the US subsidiary of UK-based Polyhedra, Plc.*