

# In-Memory Database Web Server

*The Internet Appliance model places new demands on IP networking and web-server resources. On the Web client side, designers that require local data access and manipulation embed standards-based small footprint databases into a growing variety of new information appliances and Internet-enabled devices. Embedded databases will play a major role in the growth of the IP networking from both sides of the Network Appliance Model. Increasing bandwidth and the emerging wireless IP standards provide enough of an indication to see that new and advanced database technologies will be necessary to realize the potential of Network Appliances*

## INTRODUCTION

**O**n the server side, conventional databases are used to provide central control, access management, and real-time data exchange. The more web-enabled devices that are connected, via wire or wireless, the more power and scalability the database server residing at the web server must deliver.

Access demands will continue to increase as a direct result of Metcalf's Law, which states that bandwidth capacity doubles every 12 months. Increased access requires new levels of performance and reliability from the database on the web server side. The infrastructure required to continuously support higher connection levels and intensive database processing using conventional database systems can seem like a luxury to many IT professionals. However, when the success of mission-critical e-business systems depends on it, optimal database server performance with fault tolerant reliability is more of a requirement than a luxury.

In-Memory Databases (IMDB) provide an embedded database solution for reducing the cost of performance and scalable in mission-critical web/database servers. A new data storage technology, IMDB's enable high performance by removing all disk-related processing (except for the minimum amount required to provide recovery from system failures) and instead managing all data totally within available main memory.

By structuring data access for memory instead of disk, very substantial performance improvements can be achieved over a conventional Relational Database Management System (RDBMS), even when the RDBMS is able to cache the entire database within memory. Even when running all in memory, a traditional database must still execute and manage movements of data, define block sizes, and carry much of the same processing overhead as if data was residing on disk.

The potential applications for IMDB's include most aspects of traditional IT as well as many new requirements and other services that previously could not be delivered in a cost-effective manner.

Other factors are driving the adoption of IMDB technology. Moore's Law, the decreasing price of Random Access Memory, and now 64-bit technology are prompting IT professionals to ask "why not put the

whole Web database in memory?" Even if only used selectively for pages with high-traffic and heavy query resolution, or requiring real-time transaction processing, an IMDB can significantly enhance both performance and quality of service.

## A REAL-TIME APPLICATION SERVICE PROVIDER

Polyhedra is an object-relational IMDB that is currently embedded into a variety of commercial applications worldwide. With the active object-relational nature of a Polyhedra database, new and more powerful approaches to building web database servers and providing application services are possible.

Responsiveness goes up a notch when Web clients access data and processes residing and running in main memory. This is even more obvious when the immediate task is to process data to build dynamic pages for multiple requests at once. However, beyond that, you can gain even greater efficiency by eliminating the Web server layer between the Web client and the database.

```
create schema
  create table data
  (
    persistent
    , name      char primary key
    , type      char virtual
    , value     binary
    , timestamp datetime
    , textvalue char -- (easy way of setting value)
  )
  create table html
  (
    persistent
    , derived from data
  )
  create table graphic
  (
    persistent
    , derived from data
  )
  create table gif
  (
    persistent
    , derived from graphic
  )
  create table appl_data
  (
    persistent
    , derived from data
  );
```

Sample code 1.

# SOFTWARE COMPONENTS

By encapsulating behavior, object methods in Polyhedra can be used to provide the basic functionality of a Web Server so you can eliminate this layer of 3rd party software altogether. Object methods are designed using CL, a simple object method scripting language that is part of the Polyhedra product. This platform independent language allows the database designer to program the behavior of the database, and is powerful enough to claim sockets and react to data arriving on those sockets. The simplest approach defines an HTTP object method in the database that allows each table in the database to interpret and respond to HTTP messages. The database can then act directly upon HTTP messages and provide real-time responses to web clients with dynamically created pages. Polyhedra object methods can be based on any IP protocol and individual tables (classes) can contain more specific object methods. Online database applications could include dedicated HTTP-based web servers, WAP servers, NNTP news servers, remote boot servers, and even SMTP mail generation and handling.

As a Web database, Polyhedra provides SQL with extensions to support inheritance, data persistence, and more. A table structure for a simple Web site could be defined as in Sample code 1;

The database can hold data and web content and be programmed to react to the "Get" command from a connected web browser. Having received the get command and the page name, the page and data may be retrieved from the database and passed directly back to the browser.

The code that is triggered by the arrival of the HTTP

message from the web browser is able to interpret that message before responding. Thus, if the browser requests a specific .htm page that must be built dynamically from the data in the database, that page can be recognized and built by the database itself, before being passed back to the browser.

Similarly, where the browser returns the data entered into a form, the code running within the database can interpret the returned data and use that data to build an appropriate page to return to the browser, or perhaps pass the details to a third party database for billing purposes, or maybe search the database of pages and return search results.

## **The function that actually decides what page to return**

Respond is called by the HTTP classes within the database when a page request has been recognized. Respond will take the variable "Request" and analyze it for which page is required. In this simple example, if the page requested is "dynamic\_display" or "search\_results", then a function is invoked to build those pages. Otherwise the function "GetPage" is invoked to get the desired page from the database and return it to the browser. (See Sample code 2)

The possibilities of the highly programmable and scripted database engine are limited only by the designers imagination. Polyhedra offers the database designer a hitherto impossible variety of possibilities. The database is not a mere static repository of data and pages but is responding to HTTP requests from the web browser intelligently, and with the tight integration to the Web.

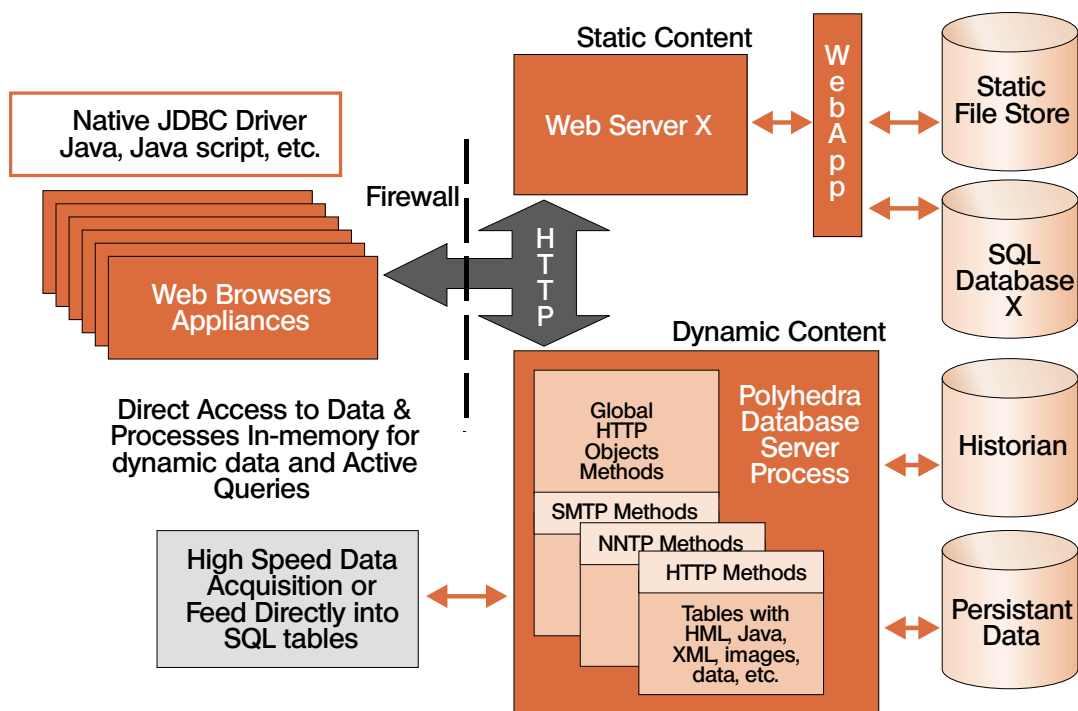


Figure 1. Polyhedra encapsulates everything necessary to provide database access from the Web.

```
function binary Respond
-- analyze request
    local char pagename
    if lowercase (word 1 of request) = "get" then
-- It's a "get" command, followed by a page name
        set pagename to word 2 of request
-- Test if page name is the dynamic page, or the search
results page. If so, then get the functions that handle those
pages to build the page, otherwise, just get the page from
the database. --
        if pagename = "dynamic_display" then
            return Dynamic_Page(pagename)
        else if pagename = "search_results" then
            return Search_Results(pagename)
        else
            return GetPage (word 2 of request, 200)
        end if
    else
        return GetPage ("/400.txt", 400)
-- Return 400 code page
    end if
end Respond
```

Sample code 2.

To illustrate how this can be done, and to simplify the task for web database developers, a sample set of HTTP web server support object methods are included in the standard Polyhedra demo package.

Adding Java applets allows the web designer yet another level of integration with the database. Now, the Java programmer is able to get tightly coupled access to the web database, and may take advantage of the Active Query mechanism to push changing data to the applet in real-time. The Type 4 JDBC driver, fully written in Java and taking advantage of the Polyhedra client-server protocol for the most efficient access to the database, can access the desired data in the database to generate a highly responsive and functional interface for the web browser client. The access to data in the database through the JDBC driver is controlled via a built in user-security mechanism. Users may be granted and revoked various access rights to the data, in a mechanism that is secure and integrated completely into the database. The Java programmer may create any kind of front end to the data, and may allow the user to modify the data in the database directly where appropriate for the tightest coupling between the end user display and the data held in the Polyhedra database.

Polyhedra's Active Query mechanism provides a fine-grained SQL Push technology. Active Queries allow SQL statements to be persistent at the database server. Until the client closes the query, it automatically receives any changes (deltas) in the original result set and has complete control over the delivery of the data. Web clients accessing a Polyhedra web/database server receive these "deltas" dynamically in real-time, including all individual attribute changes of any affected records.

Using an Active SQL Query from, say, a Java Applet, you can set up a browser client to continuously receive any changes of a table within Polyhedra, for example, the movement of a stock price or change in status of another object. Active SQL Query ensures that connected browsers never need to poll the server or re-issue the query in order to keep up with changes in real-time.

## 24X7 AVAILABILITY

Database systems supporting current and next-generation wireless and broadband network infrastructure demand the same reliability as proprietary systems in the traditional telecom and utilities industries. To meet this challenge, IMDBs must provide a solution for Web transaction database service on a 24x7 continuous basis using standard hardware and software components without the need for proprietary application development.

Polyhedra fault tolerance provides the highest level of availability to ensure Web client transactions are executed without interruption. It does this by providing a complete and flexible set of functionality for instant fail-over from master database service to a replicated standby database server and maintains client application connections as well as transactional reliability. Fail-over is imperceptible to clients and happens at machine speed, typically less than 100 milliseconds, in the event of a controlled switch-over or a failure of the master database.

Embedded databases will play a major role in the growth of the IP networking from both sides of the Network Appliance Model. Increasing bandwidth and the emerging wireless IP standards provide enough of an indication to see that new and advanced database technologies will be necessary to realize the potential of Network Appliances ■

---

*Dave Morse is Marketing Director & Managing Director at Polyhedra, USA. Mr. Morse has been a business and technical marketing manager in the database systems industry for over 15 years. Having started and sold a database development tools company and leading a successful marketing management career with a major embedded database vendor, Mr. Morse provides extensive experience in the embedded database market. Mr. Morse earned a BS in Management from University of Oregon in 1981.*



**Dedicated Systems**  
Encyclopaedia

<http://www.dedicated-systems.com>