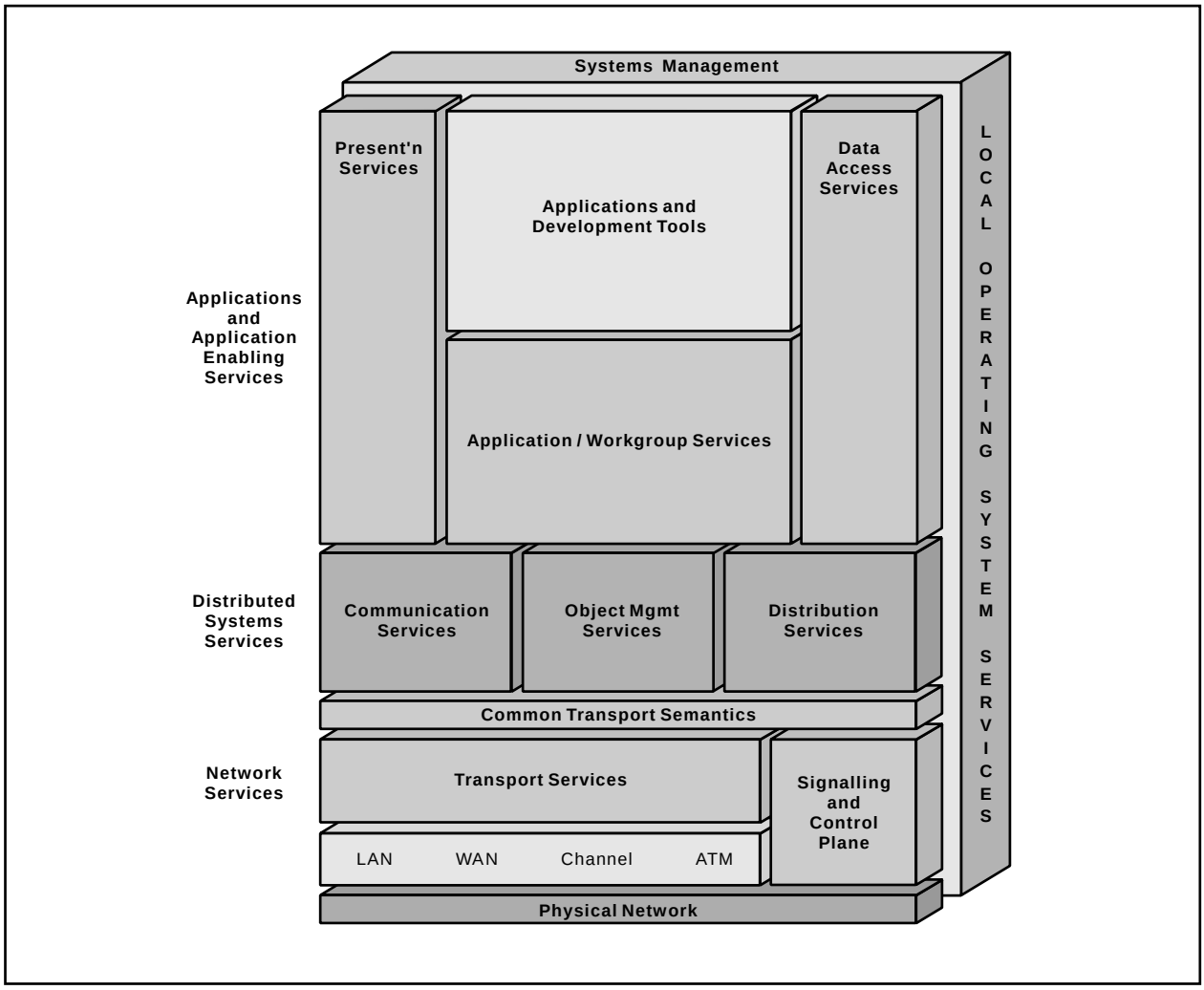
Open Blueprint

Object-Oriented Database Resource Manager



GC23-3922-01

Open Blueprint

Object-Oriented Database Resource Manager

GC23-3922-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Object-Oriented Database resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Object-Oriented Database Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Summary Of Changes

This revision describes the ability to access the Object-Oriented Database resource manager from the World Wide Web, and includes information about the use of Java applets.

Open Blueprint Object-Oriented Database Resource Manager

The Object-Oriented Database resource manager (OODBRM) provides object-oriented database facilities to applications and resource managers in IBM and non-IBM environments using standard and proprietary application programming interfaces (APIs). This paper gives an overview of the OODBRM and describes its role in the Open Blueprint.

User Functions

The OODBRM provides facilities for creating, storing, and accessing objects in an object database. The stored objects are specified by programming language constructs and are operated on in application programs. The object data model used by an application is based on the language in which it is programmed. The OODBRM takes advantage of the data models of such powerful languages as C++, Java, and Smalltalk, and provides an application programming interface to the OODBRM runtime system for each of them. The application program treats persistent objects as it does transient objects, except that it must open the object database and place the creation, access and modification of persistent objects within transaction boundaries. Because access to persistent data is typically as fast as access to transient data, the OODBRM is especially suitable for applications that must navigate very rapidly over complex objects, that is, objects that contain references (or pointers) to other objects.

Data of any C++, Java, Visual Basic, or Smalltalk data type can be allocated transiently (in dynamic virtual memory) or persistently (in a database). Data objects can consist of simple built-in language types, such as integers and character strings, or of complex class types containing embedded structures and pointers. The application operates on persistent data within the scope of a transaction. At the end of a transaction, the application chooses either to commit any data updates or discard them. The OODBRM ensures that all persistent data resides on non-volatile storage following a successful transaction commit. When a transaction aborts, none of the objects updated during the transaction are written to the database but instead are restored to their state prior to the beginning of the transaction.

Data Modeling Facilities

The OODBRM allows the application developer to use the full power of object-oriented programming languages to define data objects. Language features, such as inheritance, polymorphism, and encapsulation, are fully supported. The OODBRM provides facilities for modeling objects and managing large sets of objects. These facilities include collections, queries on collections, relationships between objects, and versioning.

Collections are abstract structures that resemble arrays in programming languages or tables in a relational database. Performance of the OODBRM collections facility is comparable to that of hand-coded, low-level data-structures, such as b-trees or hash tables. The collection classes provided by the OODBRM have methods for inserting, removing, and retrieving elements of a given collection, and for the set-theoretic union, intersection, and subset operations.

Collections are primarily used in database queries. A query produces a collection of database object references that satisfy the query expression. The query is not an SQL statement, as is a relational database query, but rather an expression in the programming language used for writing the application. However, like a relational DBMS, the OODBRM provides indexes to improve data retrieval performance.

When modeling complex objects such as documents and parts hierarchies, relationships are often used. A *relationship* is composed of two or more objects that are constrained to be consistent with one another

in a manner specified by the programmer. For example, a relationship exists between a book document and its chapters. The OODBRM can automatically enforce referential integrity between two related objects.

The OODBRM provides the ability to store and manipulate multiple versions of an object simultaneously. Known as *versioning*, this function is useful for modelling many activities, such as writing a document, where iterative processes require experimentation with various modifications.

Client/Server Architecture

The OODBRM supports cooperative access through its client/server architecture. The goal of distributed database architecture is to enable clients to have transparent access to objects, regardless of location or storage format, and to allow client and server processes to run on different types of computers and networks. An OODBRM server can support a client of any type regardless of data format, byte ordering, floating-point representation, or data alignment. Databases created by a client on one machine can be accessed by a client on a machine with a totally different processor architecture.

The server part of the OODBRM manages the physical data of one or more databases and arbitrates among client processes making requests for the data. A particular database is managed by a specific server. The server communicates with clients and provides transaction control, buffer management, checkpointing, restart, and recovery. A single server can support many clients running on the same or different machines. A single client can simultaneously access multiple databases on one or more servers.

The client part of the OODBRM supports application programs that access object databases. The client manages the logical view of the data, which includes collections, queries, relationships, transaction management, and memory management. In the OODBRM, much of the query and data management processing occurs on the client side of the network, not on the server side.

The server and the client communicate through a network when they are running on different hosts and through local operating system facilities, such as shared memory and sockets, when they are running on the same host. Several network protocols are used to transport data including TCP/IP, APPC, NetBIOS, and IPX/SPX. The client and the server interact through established protocols using private inter-process function protocols or distributed object management services consistent with those defined by the Open Blueprint.

Concurrency Control

The OODBRM server provides the serialization and concurrency control needed to allow multiple clients to access the database. This control is accomplished using lock management. Locks on data for multiple user concurrency control are managed automatically and transparently. This locking model ensures data integrity by managing a transaction's read and write sets. To minimize the need for network communication, data locking information is available to both the database client and the database server.

A copy of an object in the client address space is marked with either a shared or an exclusive access mode. The server is aware of which locks are held by which clients and with which modes. When a client requests an object from the server and the server notices that the object is in the cache of another client, the server will check to see if the modes conflict. If modes conflict, the client requesting the lock must wait for it to free. If the locks do not conflict, the server allows concurrent access. Standard locking protocols are implemented when many applications are permitted to read data concurrently but only one client process is permitted to update a particular object within a transaction.

Another feature of the OODBRM that enhances concurrent database access is multi-version concurrency control. This feature enables applications to open a database in a read-only mode that provides a transaction-consistent view of the data, without lock conflicts, even though another process may begin and commit an update transaction. Any object that has been updated during a transaction will be written to the database when the transaction commits.

Virtual Memory Mapping

For an OODBRM that manages complex objects, the speed of referencing objects is the most important factor affecting performance. During an application session, referenced persistent data is dynamically mapped into the application's virtual address space. This mapping ensures that access to persistent data is exactly as fast as access to transient data, once the object has been sent to the client and placed in its virtual memory.

The OODBRM takes advantage of the processor's memory paging hardware and the local operating system interfaces that allow allocation in and reclamation of virtual address space. When an application dereferences a pointer whose target has not yet been retrieved into the client's address space, the hardware detects an access violation and raises an exception. The OODBRM client handles this exception by retrieving the object from the database and placing it in the application's virtual memory. It then returns to the application process, which resumes processing the referenced object. Subsequent references to that object, or to other objects that have been mapped into memory, will execute in a single instruction without causing a memory fault.

Internet and Intranet Access

The OODBRM provides database technology for Internet and intranet access that can be used in the delivery infrastructure of the World Wide Web (Web). There are tools that assist developers in creating dynamic and interactive Web applications and libraries that facilitate the definition, storage, management, and delivery of Web objects.

HyperText Markup Language (HTML) templates containing tags designed for object database access automate the processing and presentation of Web application objects. An OODBRM tool reads customized templates to determine how to query or update the database, then dynamically generates HTML to display the results. This HTML is sent directly to the HyperText Transfer Protocol (HTTP) resource manager (Web server). The Web server sends the HTML on to a Web browser. Thus, extended HTML tags control the presentation and customization of WWW applications involving the OODBRM.

The OODBRM provides library support for extended types used in Web applications, such as HTML, video, audio, image, and Java applets. These types of objects can be defined to the OODBRM and can be stored either as OODBRM data or as associated file system data. For example, a Java applet used in a customized HTML page can be stored in an object database along with its associated metadata, which includes keyword, author, and version.

Gateway to other DBMSs

The OODBRM facilitates transparent inter-operability among various database management systems by means of a gateway. The gateway provides access directly to relational data, such as the data managed by the DB2 Family, Sybase, and Oracle, and indirectly to non-relational data such as IMS DL/1 data. To facilitate transparent data manipulation from various database structures, a direct correspondence, called a *schema mapping*, is created between non-object data models and object data models. A schema mapping describes how the elements of a relational schema (table, tuple, key, foreign key) or IMS data

base definition (DBD) corresponds to analogous elements of the OODBRM schema (class, object, object Id, reference). During application execution, object data values mapped to relational database tables are read and written automatically and transparently.

Scalability

The OODBRM is capable of operating in a small environment where the resource manager server and client are resident on the same local, standalone machine. Because the OODBRM is also capable of operating as a distributed resource manager in a heterogeneous environment, it scales with respect to application complexity and database size. As an application grows in complexity, more clients can be added, possibly on different machines, to absorb the increased computational load. As the application's data requirements increase, the number of databases, managed by distributed servers, can be increased.

The Role of the OODBRM in the Open Blueprint

The OODBRM is a distributed resource manager. It supports the Open Blueprint's concepts of distributed computing by separating and encapsulating client and server functions for an open, heterogeneous, distributed system.

Object Management Services

The OODBRM supports OMG CORBA-compliant implementations on IBM and non-IBM platforms.

Federated Naming and the Open Blueprint Directory

The OODBRM will adopt the federated naming standards of the Open Blueprint. The Directory resource manager will be used by clients to locate OODBRM servers anywhere on the network.

Security

The OODBRM will use the Open Blueprint Identification and Authentication resource manager for user authentication, which will preserve the single-signon property of the network system. When a user is identified during logon, their credentials, expressed as memberships in specified groups, are established and subsequently used by the Access Control resource manager to make decisions regarding access to object databases on the network.

Transaction Manager

The OODBRM as a resource manager will participate along with other resource managers in a multiple resource update (for example, simultaneous updates to several databases) using the Open Blueprint Transaction Manager. This update will become possible when the OODBRM supports the XA interface for the two-phase commit protocol.

Platform Support

IBM has negotiated an agreement with Object Design Inc. to use its ObjectStore product as the strategic IBM OODBRM. IBM's OODBRM currently supports numerous server and client workstation and desktop platforms such as OS/2, AIX, Windows NT, SunOS, Solaris 2, HP-UX, IRIX 5, Digital Unix, and Novell Netware. These platforms can be intermixed as object database servers and as object database clients.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM' valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service. may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
DB2
IBM
IBMLink
Open Blueprint
OS/2
SOM
System Object Model

The following terms are trademarks or service marks of other companies:

C++	American Telephone and Telegraph Company, Incorporated
Corba	Object Management Group, Incorporated
Digital	Digital Equipment Corporation
HP	Hewlett-Packard Company
IPX	Novell, Incorporated
Java	Sun Microsystems, Incorporated
Novell	Novell, Incorporated
ObjectStore	Object Design, Incorporated
Open VMS	Digital Equipment Corporation
Solaris	Sun Microsystems, Incorporated
SunOS	Sun Microsystems, Incorporated
Sybase	Sybase Incorporated
Visual Basic	Microsoft Corporation
Windows	Microsoft Corporation
Windows NT	Microsoft Corporation

Microsoft, Windows and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3922-01