Bringing Object-Relational Technology to The Mainstream

Vishu Krishnamurthy
Oracle Corporation
500, Oracle Parkway,
Redwood Shores, CA 94065
1-650-506-0594
vkrishna@us.oracle.com

Sandeepan Banerjee
Oracle Corporation
500, Oracle Parkway,
Redwood Shores, CA 94065
1-650-506-2817
sabanerj@us.oracle.com

Anil Nori
Ventis Corporation
University Avenue,
Palo Alto, CA
1-650-330-3082
anori@ventis.com

ABSTRACT

Over the last few years, Oracle has evolved its flagship relational database system into an Object-Relational system by adding an extensible type system, object storage, an object cache, an extensible query and indexing framework, support for multimedia datatypes, a server-based scalable Java virtual machine, as well as enhancing its SQL DDL and DML language. These extensions were done with the practical goal of bringing objects to mainstream use.

Keywords

Data Cartridges, Extensibility, *i*FS, *inter*Media, multimedia, Object-Relational, SQL3, AQ.

1. INTRODUCTION

The new Internet computing environment has suddenly brought new kinds of data to thousands of users across the globe. Multimedia data types like images, maps, video clips, and audio clips were once rarely seen outside of specialty software. Today, many Web-based applications require their database servers to manage such data. Other software solutions need to store data dealing with financial instruments, engineering diagrams, or molecular structures.

With the addition of object-relational extensions, the Oracle8*i*™ server can be enhanced by developers to create their own application-domain-specific data types. For example, you can create new data types representing customers, financial portfolios, photographs or telephone networks – and thus ensure that your database programs deal with the same level of abstraction as your application domain. In many cases, it is desirable to integrate these new domain types as closely as possible with the server so they are treated at par with the built-in types like NUMBER or VARCHAR. With such integration, the database server can be readily extended for new domains.

2. ORACLE'S OBJECT-RELATIONAL TECHNOLOGY

2.1 EXTENSIBILITY FRAMEWORK

Normally, the database provides a set of services for example the basic storage service, a query processing service, services for indexing, query optimization and so on. Applications use these services to avail themselves of database functionality.

In Oracle8*i*, these services are made extensible so that data cartridges can provide their own implementations of these services. When some aspect of a native service provided by the database is not adequate for the specialized processing that a developer may provide a domain-specific implementation. For example, if you build a Spatial Data Cartridge for Geographical Information Systems, you may need the capability to create spatial indexes. To do this, you would implement routines that create a spatial index, insert an entry into the index, update the index, delete from it, and so on. The server would automatically invoke your implementation every time indexing functionality was needed for spatial data. In effect, you would have extended the Indexing Service of the server.

The programming interfaces that enable extending the various services are collectively called the Extensibility Framework.

This framework enables the creation of server-based components called Data Cartridges. For example, a spatial data cartridge may provide comprehensive functionality for a geographic domain such as being able to store spatial data, perform proximity/overlap comparisons on such data, and also integrate spatial data with the server by providing the ability to index on such data. Data cartridges can have broad horizontal utility (such as imaging, time series, spatial data) or narrow vertical focus (such as piping diagrams, networks or bank accounts). The data cartridge programming-interface is publicly available and is also used by *inter*Media (see below).

2.2 OBJECT TYPE SYSTEM

Historically, applications have focused on accessing and modifying corporate data that is stored in tables composed of native SQL data types such as INTEGER, NUMBER, DATE, and CHAR. In Oracle8*i*, there is support not only for these native types, but also for new 'object' data types, tracking the semantics of the forthcoming SQL3 standard.

2.2.1 Object Types

An object type, distinct from native SQL data types, is user-defined and specifies both the underlying persistent data (called 'attributes' of the object type) and the related behaviors ('methods' of the object type). Object types are used to extend the modeling capabilities provided by the native data types. You can use object types to make better models of complex entities in the real world by binding data attributes to semantic behaviors.

A method is a procedure or a function that is part of an object type definition. Methods can be run within the execution environment of Oracle8*i* or dispatched to run outside it.

2.2.2 Collection Types

Collections are SQL data types that contain multiple elements. Each element or value for a collection is the same data type. In Oracle8*i* there are two collection types – VARRAYs and Nested

Tables. A VARRAY contains a variable number of ordered elements. VARRAY data types can be used as a column of a table or as an attribute of an object type.

Using Oracle8*i* SQL, a named table type can be created. These can be used as Nested Tables to provide the semantics of an unordered collection. As with VARRAY, a Nested Table type can be used as a column of a table or as an attribute of an object type.

2.2.3 Relationship Types

If you create an object table in Oracle8*i*, it is possible to obtain a reference (or the database po*inter*) to an associated row object. References are important for modeling relationships and navigating among object instances particularly in client-side applications.

2.2.4 Large Objects

Oracle8*i* provides the large object (LOB) types to handle the storage demands of images, video clips, documents, and other such forms of data. Large objects are stored in a manner that optimizes space utilization and provides efficient access. More specifically, large objects are composed of locators and the related binary or character data. The LOB locators are stored in-line with other table record columns and for *inter*nal LOBs (BLOB, CLOB, and NCLOB) the data can reside in a separate storage area. For external LOBs (BFILEs), the data is stored outside the database tablespaces in operating system files.

3. JAVA SUPPORT

Oracle provides native support for Java in the DBMS. That is, Oracle has developed its own Java VM that integrates with the database mechanisms closely for high performance and scalability. In addition, the database system natively supports JDBC and SQLJ, an ORB and the EJB framework. The Oracle8*i* also comes with a web server; so, the database system can be an HTTP listner.

The object relational facilities provide a more natural and productive way to maintain a consistent structure between a set of Java classes in at the application level and the data model at the data storage level. With the integration of Java with the Oracle8*i* server, Oracle provides a tight integration between its object-relational facilities and Java. It does so in three important ways:

- Java class instances can be stored as Oracle object type instances.
- Server Object-Relational schema can be mapped to Java classes**.**
- Java is an implementation choice for object type methods.

4. OBJECT CACHE

Oracle8*i* comes with a client-side cache for object-relational data to give applications the following benefits:

- Transparent mapping of database objects to host language objects in memory, and memory management for such objects with full transactional semantics.
- Navigational object access when operating on a graph of inter-connected objects.
- The ability to operate on the transitive closure of the root object in one network roundtrip.
- The object cache supports both a pessimistic locking scheme and an optimistic locking scheme.

5. ORACLE PRODUCTS BASED ON OBJECT-RELATIONAL TECHNOLOGY

5.1 *inter*Media

Internet-based applications require advanced data management services that support the rich data types used in electronic commerce catalogs, corporate repositories, and other media-rich applications. Oracle *inter*Media extends Oracle8*i* reliability and availability to applications that need access to image, audio, video, location, text, and relational data.

Open APIs enable standard SQL access using native text, image, audio, and video, and other data type services, operators, and metadata management, allowing third parties to build additional media extensions for specialized data types.

*Inter*media is tightly integrated with the Oracle database system via the extensibility framework.

5.2 INTERNET FILE SYSTEM (*i*FS)

The Oracle Internet File System (*i*FS) enables the database to be treated as it were simply a shared network drive. This means that users can store and retrieve files managed by the database as if they were files managed by a file server.

The *i*FS makes it possible to view relational data stored in database tables as if they were files and folders managed by a networked drive. It allows end users to access files and folders in the *i*FS variety of protocols, giving users universal access to their data. This means that an *i*FS user seems the same files and folders from Windows Explorer, a browser, an email client or an FTP client.

The *i*FS uses the object-relational framework directly and indirectly through *inter*Media to index documents and facilitate searching and querying of these documents.

5.3 ADVANCED QUEUES

Integrating applications is an extremely complex and challenging task exacerbated by the fact that enterprise applications are geographically widely distributed, autonomously developed, and heterogeneous. Message queuing provides three essential features for the integration of such applications; an asynchronous communication model, reliable communication in the face of network failures, and a loosely coupled transaction model.

Oracle's Advanced Queuing employs the object-relational framework to store, query and project information from messages that are essentially treated as opaque objects.

6. ACKNOWLEDGMENTS

Our thanks to Andy Mendelsohn, Kenneth Ng, Chin-Heng Hong, *i*FS group, AQ group, *inter*Media group, and the SQL, Objects and Extensibility group for bringing the object-relational technology to mainstream applications.