Object-Relational Database Systems - The Road Ahead

by Ramakanth S. Devarakonda

Several major software companies including IBM, Informix, Microsoft, Oracle, and Sybase have all released object-relational versions of their products. These companies are promoting a new, extended version of relational database technology called **object-relational database management systems** also known as **ORDBMSs**. This article compares and contrasts this new class of database with the relational databases, RDBMS from which they are evolving and also with efficient object-oriented databases, OODBMSs, also known as object databases, ODBMSs.

Recently, many companies have begun using their database systems for non-traditional applications because of demands such as storing images and multimedia objects in the database. Consequently, the objects and related operations are becoming more complex. Some examples of complex data are images, geographical information systems, multimedia objects, and spatial, 3-D, and temporal data. But what are the requirements for database systems to support complex applications? Does a database supporting complex applications have to be object-oriented? A certain group thinks that future applications can only be implemented with pure object-oriented systems. Initially these systems looked promising. However, they have been unable to live up to the expectations. A new technology has evolved in which relational and object-oriented concepts have been combined or merged. These systems are called object-relational database systems. The main advantages of ORDBMSs are massive scalability and support for object-oriented features.

Relational DBMS (RDBMS)

The relational model was formally introduced by Dr. E. F. Codd in 1970 [2] and has evolved since then, through a series of writings and later through implementations by IBM and others. The defining standard for relational databases is published by ANSI (the American National Standard Institute) as SQL (ANSI 1986) or SQL1, called SQL-86. A revised standard is called SQL2, also referred to as SQL-92.

A relational database is composed of many relations in the form of two-dimensional tables of rows and columns containing related tuples. Organizing data into tables, the form in which data is presented to the user and the programmer, is known as the **logical view** of the database. The stored data on a computer disk system is called the **internal view**. The rows (tuples) are called **records** and the columns (fields in the record) are called **attributes**. Each column has a data type (i.e., int, float, date). There are various restrictions on the data that can be stored in a relational database. These are called **constraints**. The constraints are domain constraints, key constraints, entity integrity constraints, and referential integrity constraints. These constraints ensure that there are no ambiguous tuples in the database.

RDBMSs use Structured Query Language (SQL, currently SQL2) as the data definition language (DDL) and the data manipulation language (DML). SQL includes statements for data definition, modification, querying and constraint specification. The types of queries vary from simple single-table queries to complicated multi-table queries involving joins, nesting, set union/differences, and others. All processing is based on values in fields of records. Examples of RDBMSs include Oracle, developed by Oracle Corporation, and Microsoft Access developed by Microsoft. The main disadvantages of Relational Databases include their inability to handle application areas like spatial databases (e.g. CAD), applications involving images, special types databases (e.g. complex numbers, arrays, etc.) and other applications that involve complex interrelationships of data. The SQL standard enables users to easily migrate their database applications between database systems. In addition, users can access data stored in two or more RDBMSs without changing the database sub-language (SQL). The other merits include rapid data access and large storage capacity [3].

Object-Oriented DBMS (OODBMS)

The desire to represent complex objects has led to the development of object-oriented (OO) systems. The concept of **abstract data types (ADTs)** in which the internal data structure is hidden and the external operations can be applied on the object that is specified led to the concept of encapsulation. The programming language SMALLTALK, developed by Xerox, was explicitly designed to be object-oriented. Other object-oriented programming languages include C++, Java, etc. The main features of OO programming languages are encapsulation, inheritance and polymorphism. **Encapsulation** can be thought of as a protective layer that prevents the code and the data from being accessed by other code defined outside the layer. The process in which one object inherits the properties of a previously defined object is called **inheritance**. Inheritance aids in the reuse of existing definitions for creating new objects. **Polymorphism** allows the same operator or symbol to have different implementations, depending on the type of objects to which the operator is applied.

Object-oriented databases employ a data model that supports object-oriented features discussed above and abstract data types. OO databases provide unique object identifiers (OIDs) so that the objects can be easily identified. This is similar to a primary key in the relational model. Object-oriented databases utilize the power of object-oriented programming languages to provide excellent database programming capability. The data in object-oriented database management systems (OODBMSs) is managed through two sets of relations, one describing the interrelations of data items and another describing the abstract relationships (inheritance). These systems employ both relation types to couple data items with procedural methods (encapsulation). As a result, a direct relationship is established between the application data model and the database data model. The strong connection between application and database results in less code, more natural data structures, and better maintainability and reusability of code. OO languages, such as C++ or Java, are able to reduce code size by not having to translate code into a database sublanguage such as SQL and ODBC or JDBC [5, 6].

Until recently, the lack of a defining standard was a drawback for OODBMSs. The Object Data Management Group (ODMG) has proposed a standard known as ODMG-93 or ODMG 1.0 standard, now revised into ODMG 2.0 [1]. The standard consists of the object model, the object defining language (ODL), the object query language (OQL), and the bindings to OO programming languages. The ODL and OQL are based on the ODMG data model. The data model consists of data types, type constructors, etc., and is similar to the SQL report that describes the standard model for relational databases. The ODL is designed so as to support semantic constructs of ODMG 2.0 object model. It is independent of any programming language. The ODL is used to create object specifications. The OQL is designed to work closely with the programming languages for which an ODMG binding is defined such as C++, Java and SMALLTALK. The syntax of the OQL queries is similar to the syntax of SQL (a query language for relational databases) with some additional features such as object identity, complex objects, inheritance, polymorphism and relationships. An object-oriented language is the language for both the application and the database. OODBMSs have been integrated with C++, C, Java and LISP. The primary interface in an OODBMS for creating and modifying objects is directly via the object language (C++, Java, etc.) using the native language syntax. A key difference between relational databases and OO databases is the way in which relationships are handled. In OO databases, the relationships are represented explicitly with OIDs, which improves the data access performance. In relational databases, relationships among tuples are specified by attributes having the same domain.

The main drawback of OODBMSs has been poor performance. Unlike RDBMSs, query optimization for OODBMs is highly complex. OODBMSs also suffer from problems of scalability, and are unable to support large-scale systems.

Some examples of OODBMSs are O2 (now called Ardent) developed by Ardent Software, and the ObjectStore system produced by Object Design Inc. [3, 6]

Object-Relational DBMS (ORDBMS)

The main objective of ORDBMS design was to achieve the benefits of both the relational and the object models such as scalability and support for rich data types. ORDBMSs employ a data model that attempts to incorporate OO features into RDBMSs. All database information is stored in tables, but some of the tabular entries may have richer data structure, termed *abstract data types* (ADTs). An ORDBMS supports an extended form of SQL called SQL3 that is still in the development stages. The extensions are needed because ORDBMSs have to support ADT's. The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns and SQL is used as the query language and the result of a query is also table or tuples (rows). But the relational model has to be drastically modified in order to support the classic features of object-oriented programming. Hence the characteristics of an ORDBMSs are:

- Base datatype extension,
- Support complex objects,
- Inheritance, and
- Rule Systems [9].

ORDBMSs allow users to define datatypes, functions and operators. As a result, the functionality of the ORDBMSs increases along with their performance.

An example schema of a student relation which ORDBMS supports is :

STUDENT(fname,lname,ID,sex,major,address,dname,location,picture)

Notice the extra attributes "location" and "picture" which are not present in the traditional EMPLOYEE relation of RDBMS. The datatype of "location" is "geographic point" and that of "picture" is "image".

The differences between the three approaches

Table 1: A Comparison of Database Management Systems

Criteria	**RDBMS**	**ODBMS**	**ORDBMS**
Defining standard	SQL2	ODMG-2.0	SQL3 (in process)
Support for object-oriented features	Does not support; It is difficult to map program object to the database	Supports extensively	Limited support; mostly to new data types
Usage	Easy to use	OK for programmers; some SQL access for end users	Easy to use except for some extensions
Support for complex relationships	Does not support abstract datatypes	Supports a wide variety of datatypes and data with complex inter-relationships	Supports Abstract datatypes and complex relationships
Performance	Very good performance	Relatively less performance	Expected to perform very well

Product maturity	Relatively old and so very mature	This concept is few years old and so relatively mature	Still in development stage so immature.
The use of SQL	Extensive supports SQL	OQL is similar to SQL, but with additional features like Complex objects and object-oriented features.	SQL3 is being developed with OO features incorporated in it
Advantages	Its dependence on SQL, relatively simple query optimization hence good performance	It can handle all types of complex applications, reusability of code, less coding	Ability to query complex applications and ability to handle large and complex applications
Disadvantages	Inability to handle complex applications	Low performance due to complex query optimization, inability to support large-scale systems	Low performance in web applications
Support from vendors	It is considered to be highly successful so the market size is very large but many vendors are moving towards ORDBMS	Presently lacking vendor support due to vast size of RDBMS market	All major RDBMS vendors are after this so has very good future
Source: International Data Corporation, 1997 [4]			

In a paper, "Object-Relational DBMS: The Next Wave," [8] Dr. Michael Stonebraker, Chief Technology Officer of Informix Software, has classified the DBMS applications into four types: simple data without query, simple data with query, complex data without query, and complex data with query. These four types describe file systems, Relational DBMSs, Object-Oriented DBMSs, and object-Relational DBMS, respectively. Universal Server, developed by Informix, belongs to the fourth category. The other current ORDBMSs include Oracle8, from Oracle Corporation, and Universal DB (UDB) from IBM. Also, Stonebraker predicts that applications from Relational DBMSs (simple data with query) will slowly move towards the Object-Relational DBMSs (complex data with query). To explain this, he gives the example of an insurance company having a customer database and a claims database and following traditional data processing applications in Relational DBMSs. If to this application, the company wants to add the diagram of each accident site, a scanned image of the police report, the picture of dented car, the latitude and longitude of the accident site, and the latitude and longitude of each customer's house in order to estimate the validity of accident and to avoid fraudulent claim of money, then in this case the application should move from Relational DBMSs to ORDBMSs. This is why he terms ORDBMSs as "the next wave" [8].

Table 2: Categories of DBMSs

File Systems	**RDBMSs**	**OODBMSs**	**ORDBMSs**
Simple data without queries	Simple data with queries	Complex data without queries	Complex data with queries

The five architectural options given by Dr. Stonebraker, listed in ascending order of practicality, performance, and general desirability are:

- Supply plug-in code to make function calls to other applications.
- Add separate API's and server subsystems to support object functionality.
- Simulate specialized object-relational functionality in a middleware layer.

- Completely redesign the database engine.
- Add a new object-oriented layer to support rich datatypes atop a proven relational database engine [7].

The main advantage of ORDBMSs is their massive scalability. Oracle8, released by Oracle Corporation, is designed to manage large amounts of information. Oracle8 is expected to help NASDAQ manage its Very Large Data Bases, VLDBs, which contain hundreds of gigabytes of time series applications are required by traders and analysts to examine trends on stock data. In spite of many advantages, ORDBMSs also have a drawback. The architecture of object-relational model is not appropriate for high-speed web applications. However, with advantages like large storage capacity, access speed, and manipulation power of object databases, ORDBMSs are set to conquer the database market. The support from major DBMS vendors and its features will make ORDBMSs the market leader. The International Data Corporation (IDC) also expresses the opinion that the ORDBMS market will surpass the size of the ODBMS market in next three years.

References

1
Cattell, R.G. *The Object Database Standard: ODMG-93 (Release 1.2)*. Morgan Kauffman Publishers. San Francisco, 1995.

2
Codd, E.F. *A Relational Model of Data for Large Shared Data Banks.* CACM 13(6): 377-387. 1970.

3
Elmasri, R. & Navathe, S.B. *Fundamentals of Database Systems 3e*. Addison Wesley, 2000.

4
McClure, S. *IDC Bulletin #14821E*. August, 1997.

5
Nahouraii, E. & Petry, F. *Object-Oriented Databases*. IEEE, 1991.

6
Rao, B.R. *Object-Oriented Databases: Technology, Applications, and Products* McGraw-Hill. New York, 1994.

7
Stonebraker, M. *Architectural Options for Object-Relational DBMSs*. Informix Software, CA. Feb, 1997.

8
Stonebraker, M. *Object-Relational DBMS: The Next Wave*. Informix Software, CA.

9
Stonebraker, M., Brown, P., and Moore, D. *Object-Relational DBMSs: Tracking the Next Great Wave 2e*. Morgan-Kauffman Publishers. San Francisco, 1999.

Biography

Ramakanth Subrahmanya Devarakonda (ramakanthds@usa.net) is a Master's student of Computer Science at the Old Dominion University, Norfolk, VA. He is a graduate in Computer Science and Engineering from Andhra University, A. P., India.