Managing the Data Effectively Using Object Relational Data Store

T. Sivagamasundari

Research Scholar, Department of Computer Science, Prist University

Abstract: ***The collection of interrelated data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of Database Management System is to provide a way to store and retrieve database information that is both convenient and efficient. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. A*** **relational database** ***consists of a collection of relations, each of which is assigned a unique name. The relational database contains a set of objects used to store, access, and manage data. The set of objects includes tables, views, indexes, aliases, distinct types, functions, procedures, sequences, and packages. Managing the data effectively using Object Relational Data Store (ORDS) is a Object Oriented Relational Database. It provides object oriented enabled features. It ensures the safety of information stored, despite system crashes or attempts at unauthorized access. If data to be shared among several users, the system will avoid possible anomalous results. It is designed to perform very well with most typical SQL operations.***

Keywords: Relational Embeddable Database, object-relational database, Object Oriented Relational Database, applications programming interfaces

1. Introduction

ORDS is designed to support the SQL standard, and provides a very full-featured implementation. It supports strong encryption. It doesn't require a database administrator or any external configuration files. Create a database by connecting to it. It's as simple as that. We have to just include the qed.jar file in your class path and use the standard JDBC interfaces. ORDS provides robust data protection and data recovery features. Committed transactions won't be lost, even if your application crashes at an "inopportune" time. Furthermore, ORDS makes it easy to perform online backups of your database, either based on a schedule, or at a time of your choosing.

1.1 Importance Choosing

ORDS is proven as a high-performance, easy-to-use, and affordable database that gives you more flexibility than proprietary solutions. The embedded server library makes ideally suited for object oriented database needs. ORDS provides robust data protection and data recovery features. Committed transactions won't be lost, even if your application crashes at an "inopportune" time. Furthermore, ORDS makes it easy to perform online

Backups of your database either based on a schedule, or at a time of your choosing. ORDS supports strong encryption. ORDS is designed to support the SQL standard, and provides a very full-featured ORDS implementation. The embedded server library makes ideally suited for embedded database needs.

1.2 Scope of the Present Work

Object relational systems are complex data types and it needs powerful query languages and high protection for the data. This is general but some database systems blur the boundaries. For example, some object oriented database systems built around a persistent programming language are implemented on top of a relational database system. Such systems may provide lower performance than object oriented database systems built directly on a storage system, but provides some of the stronger protection guarantees of relational systems. Many object-relational database systems are built on top of existing relational database systems.

Joins:
ORDS has a very simple join plan. Tables are joined left to right, with the left table being the outer, the right table being the inner table, in a series of nested inner loop INNER JOINs wherever possible. Any kind of equijoin or join on columns will use this approach. Failing a common column, we'll resort to a cross join, which is a full cartesian product. The inner table in the cross join is iterated for every row of the outer, leading to possibly very long run times.

Concurrency:
ORDS fully supports concurrent access, while maintaining SERIALIZABLE isolation and ACID properties. ORDS's Lock Manager supports a hierarchical lock tree which uses multiple lock modes to permit multiple readers and a single writer to each database structure. Locking is performed at the table level. Table locking implies that sometimes programs will block, waiting for a table lock, if it's in use by other transactions in an "inconsistent" mode. Table locking is also (as with any two-phase locking approach) subject to deadlock. ORDS inelegantly resolves this using a configurable "lock timeout" parameter.

In general, these limitations related to concurrency are the result of a conscious design compromise: ORDS's target architecture isn't designed to maximize concurrent performance. Rather, the objective is to be small and fast for typical (i.e., single user).

2. Review of Literature

2.1 Object Relational Database

Database is a collection of information organized in such a way that a computer program can quickly select desired pieces of data. We can think of a database as an electronic filing system. Traditional databases are organized by fields, records, and files. a field is a single piece of information; a record is one complete set of fields; and a file is a collection of records.

A *relational database* is a database that can be perceived as a set of tables and can be manipulated in accordance with the relational model of data. The relational database contains a set of objects used to store, access, and manage data. The set of objects includes tables, views, indexes, aliases, distinct types, functions, procedures, sequences, and packages In object relational models extend the relational data model by providing a richer type system including complex data types and object orientation. Relational query languages, in particular sql, need to be correspondingly extended to deal with the richer type system. Such extensions attempt to preserve the relational foundations, in particular, the declarative access to data- while extending the modeling power.

2.2 Key Consideration

1) Delivering a Better "Out-of-the Box" Experience
2) Full Relational Database Functionality
3) Lower Price & Total Cost of Ownership
4) Cross-platform Portability
5) Shorter Time to Market
6) Shorter Sales Cycle
7) Superior Performance, Scalability and Reliability
8) Small Footprint
9) Ease of Use
10) Administration

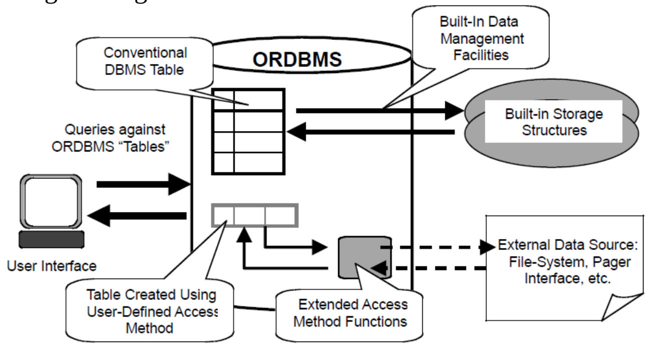
3. Methodology

ORDS consists of various processing stages. Each stage represents a level of processing the database. The Relational Embeddable Database implements SQL and JDBC 2.0.

3.1 Linking to Storage System

In order to access a database, you need to obtain a JDBC Connection object. There are two basic ways to get a database connection: Using the JDBC Driver Manager interface, you can directly obtain a JDBC Connection, if we know, the name of the JDBC Driver class (com.quadcap.jdbc.JdbcDriver). The database URL (jdbc:ORDS:database-name). Overallserver configuration information is also managed through the Config "service".

ORDBMSs possess storage manager facilities similar to RDBMSs. Disk space is taken under the control of the RDBMS, and data is written into it according to whatever administrative rules are specified. All the indexing, query processing, and cache management techniques that are part of an RDBMS are also used in an ORDBMS. Further, distributed database techniques can be adapted to incorporate user-defined types and functions. However, all of these mechanisms must be re-implemented to generalize them so that they can work for user-defined types. For example, page management is generalized to cope with variable length OPAQUE type objects. You can also integrate code into the engine to implement an entirely new storage manager.


Conventional DBMS Table
ORDBMS
Built-In Data Management Facilities
Built-in Storage Structures
Queries against ORDBMS "Tables"
User Interface
External Data Source: File-System, Pager Interface, etc.
Table Created Using User-Defined Access Method
Extended Access Method Functions

3.1 Extensible Storage Management

Performance & Administration

ORDS is designed to perform very well with most typical SQL operations. It requires zero administration. Still, sometimes you want to administer your data with ORDS, the database is simply a directory in the file system containing files accessed via a JDBC url using the ORDS JDBC driver.

3.2 Connection

The Relational Embeddable Database implements SQL92 and JDBC 2.0. Connecting to the database in order to access a database, you need to obtain a JDBC Connection object. There are two basic ways to get a database connection:
1) The name of the JDBC Driver class (com.quadcap.jdbc.JdbcDriver)
2) The database URL (jdbc:qed:*d*atabase-name)
3) Any connection parameters.

3.3 Design of the Object-Relational Database

The object-oriented methods used for the design of the systems with object-relational databases are based on the concepts of object and classes of objects and allow the use of three different models for designing an object-relational database: the static model by which are modeled objects and the relations between them; the dynamic model by which are described interactions between objects; the functional model by which are transformed data values using operations and processes.

3.4 Object-Relational Database Technology

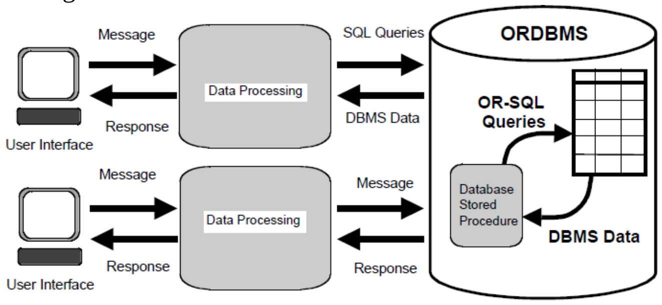
The object-relational database technology occurrence can be traced back to the middle of 1990s after emergence of object-oriented database (OODB). In their book "Object-relational DBMSs: the Next Great Wave", define their four-quadrant view (two by two matrix) of the data processing world: relational database, object-relational database, data file processing, and object-oriented database. Practically, ORDBMS bridges the gap between OODBMS and RDBMS by allowing users to take advantage of OODB'MSs great productivity and complex data type without losing their existing investment in relational data. In fact, an ORDBMS engine supports both relational and object-relational features in an integrated fashion. The u0nderlying ORDB data model is relational because object data is stored in tables or columns. ORDB designers can work with familiar tabular structures and data definition languages (DDLs) while assimilating new object-oriented features. It is essentially a relational data model with object-oriented extensions. In response to the evolutional change of ORDB technology, SQL:1999 started supporting object-relational data modeling features in database management standardization and SQL:2003 continues this evolution. Currently, all the major database vendors have upgraded their relational database products to object-relational database management systems to reflect the new SQL standards [9] and ready to be used by industrial practitioners.

3.5 ORDBMS for Object Integration

The beauty of ORDBMSs is reusability and sharing. Reusability mainly comes from storing data and methods together in object types and performing their functionality on the ORDBMS server, rather than have the methods coded separately in each application. Sharing comes from using user-defined standard data types to make the database structure more standardized.

3.6 Database Stored Procedures

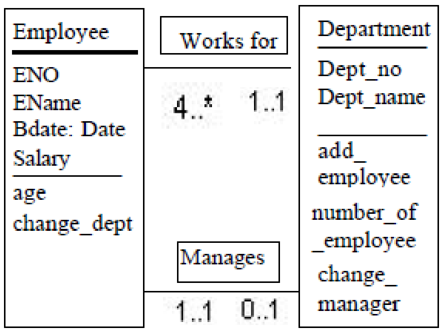
Almost all RDBMSs allow you to create database procedures that implement business processes. This allows developers to move considerable portions of an information system's total functionality into the DBMS. Although centralizing CPU and memory requirements on a single machine can limit scalability, in many situations it can improve the system's overall throughput and simplify its management.



ORDBMS as the Object Server Architecture

3.7 Desing and Implementation Tools

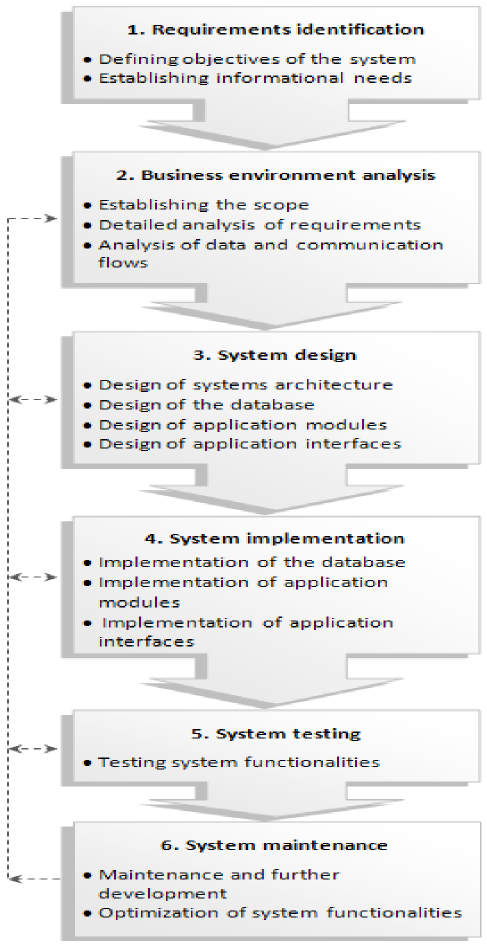
UML is used as a tool for ORDBMS design. UML is a new modeling tool developed by the Object Management Group. UML development was spearheaded by Rational Software Corp. Although the UML technology was developed mainly for software design, the important part of this technology, classes and methods, are roughly equivalent to ORDBMS types and methods. In UML class diagrams, a class is displayed as a box (see figure 1) that include three sections: the top section gives the class name; the middle section includes the attributes for individual objects of the class; and the last section includes methods that can be applied to these objects.



UML Class Diagram

4. Results and Discussions

ORDBMSs possess storage manager facilities similar to RDBMSs. Disk space is taken under the control of the ORDBMS, and data is written into it according to whatever administrative rules are specified. All the indexing, query processing, and cache management techniques that are part of an RDBMS are also used in an ORDBMS. Further, distributed database techniques can be adapted to incorporate user-defined types and functions. However, all of these mechanisms must be re-implemented to generalize them so that they can work for user-defined types. For example, page management is generalized to cope with variable length OPAQUE type objects. You can also integrate code into the engine to implement an entirely new storage manager..

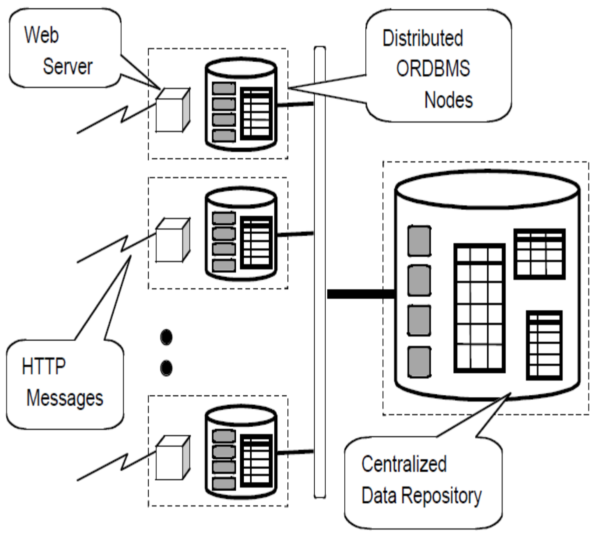

1. Requirements identification
• Defining objectives of the system
• Establishing informational needs
2. Business environment analysis
• Establishing the scope
• Detailed analysis of requirements
• Analysis of data and communication flows
3. System design
• Design of systems architecture
• Design of the database
• Design of application modules
• Design of application interfaces
4. System implementation
• Implementation of the database
• Implementation of application modules
• Implementation of application interfaces
5. System testing
• Testing system functionalities
6. System maintenance
• Maintenance and further development
• Optimization of system functionalities

Development flow of the applications with object-relational databases

4.1 Distributed Deployment

Often the volume of data in a single information system, or the workload imposed by its users, is too much for any one computer. Storing shared data, and providing efficient access to it, requires that the system be partitioned or distributed across several machines. Combining extensibility with distributed database features makes new system architectures possible. A large central machine contains canonical copies of all data. Surrounding it is a cloud of other, smaller installations.

An ORDBMS's system catalogs become a metadata repository that records information about the modules of programming logic integrated into the ORDBMS. Over time, as new functionality is added to the application and as the programming staff changes, the system's catalogs can be used to determine the extent of the current system's functionality and how it all fits together.



Distributed Information System Deployment

5. Summary and Conclusions

5.1 Summary

In spite of many advantages, ORDBMSs also had drawbacks. 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. In summary, relational and object-oriented database systems each have certain strengths as well as certain weaknesses. In general, the weakness of one type of system tends to be strength of the other.

The contribution Object relational data store uniquely provides guidelines on how to use ORDBMS to overcome relational database existing problems and improve database performance in the database development using ORDBMS features. There is some research that has been done in ORDBMS technology as ORDBMSs have become commonplace in recent years. So far very little research has been done in using ORDBMS to overcome relational database weaknesses and solve some existing normalization problems. This paper provides the guidelines for the traditional relational database practitioners to solve existing problems using ORDB technology. Many traditional database practitioners consider the ORDBMS technology as complex results in the loss of the essential simplicity and purity of the relational database model and stay away from it. There is a need to provide these professionals with the guidelines for their specific use for their future database development. This paper presents the script templates for them to implement ORDB technology in their career.

We find that the benefits of the object-oriented methods in comparison with the structured one, recommend the object-oriented approach in the case of object-relational databases design. Since object-oriented methodologies and methods have some limitations as well as many differences (in terms of symbols, notations or types of diagrams), it was needed a standard for modeling that can be widely applied in creating new systems or the maintenance of systems.

5.2 Conclusion

ORDS provides object oriented features, robust data protection and data recovery features. Committed transactions won't be lost, even if your application crashes at an "inopportune" time. Furthermore, ORDS makes it easy to perform online backups of your database, either based on a schedule, or at a time of your choosing. ORDS is proven as a high-performance, easy-to-use, and affordable database that gives you more flexibility than proprietary solutions. The embedded server library makes ideally suited for embedded database needs.

Although the user-defined methods are defined with object data within the object type, they can be shared and reused in multiple database application programs. This can result in improved operational efficiency for the IT department, as well, by improving communication and cooperation between applications. An object-relational database schema consists of a number of related tables that forms connected user-defined object-types. Object-types possess all the properties of a class, data abstraction, encapsulation, inheritance and polymorphism. These traits of object-types are embedded in the relational nature of the database; data model, security, concurrency, normalization. In more precise words, the underlying ORDB data model is relational because object data is stored in tables or columns.

The Destination Sequenced Distance Vector (DSDV) protocol is a proactive routing protocol based upon the distributed Bellman Ford algorithm . In this routing protocol, each mobile host maintains a table consisting of the next-hop neighbor and the distance to the destination in terms of number of hops. It uses sequence numbers for the destination nodes to determine “freshness” of a particular route, in order to avoid any short or longlived routing loops. If two routes have the same sequence number, the one with smaller distance metric is advertised. The sequence number is incremented upon every update sent by the host. All the hosts periodically broadcast their tables to their neighboring nodes in order to maintain an updated view of the network.

References

Book References

[1] Almeida, V.T., Güting, R.H., & Behr, T. (2006). Querying moving objects in secondo. In *Proceedings of the 7th International Conference on Mobile Data Management*.

[2] Becker, L., Blunck, H., Hinrichs, K., & Vahrenhold., J. (2004). A framework for representing moving objects. In *Proceedings of DEXA*, (pp. 854-863).

[3] Cattel, R.G.G., & Barry, D.K. (eds.). (1997, 05) *The object database Standard: ODMG 2.0.* Morgan Kaufmann Publishers.

[4] Dieker S., & Güting, R.H. (2000). Plug and play with query algebras: Secondo. A generic dbms development environment. In *Proceedings of Int'l Symp. on Database Engineering and Applications* (IDEAS), (pp. 380-390).

[5] Düntgen, C., Behr, T., & Güting, R.H. (2009). Berlinmod: a benchmark for moving object databases. *The VLDB Journal*, *18*(6), 1335-1368.

[6] Frentzos, E., Pelekis, N., Ntoutsi, I, & Theodoridis, Y. (2008). Trajectory database systems, In F. Giannotti and D. Pedreschi (eds), *Mobility, Data Mining and Privacy*. Springer.