HANDLE THE DATA EFFECTIVELY USING OBJECT RELATIONAL DATABASE MANAGEMENT

T.SIVAGAMASUNDARI[1], MR. R.SELVA KUMAR[2]

[1] *student, Department of Computer Science, Prist University, Tamil Nadu, India*
[2] *Assistant Professor, Department of Computer Science, Prist University, Tamil Nadu, India*

ABSTRACT

Object relational databases are a hybrid type of databases, which use the best facilities of its predecessors (relational and object-oriented databases) . In other words, they can be considered an object-oriented extension of the relational databases. The internal logic of storing and retrieving data is the same like in the relational case. The main difference consists in new data types, some of them user defined (like object classes), and in the ability of manipulate them.Object-relational databases, is how to store objects using tables and how to transform complex requirements of applications into properties stored in databases, all in a simple and clear way, that keep the structure of object- oriented application, reduce programming effort and maintain a reasonable level of performance. the object-relational database management system (ORDBMS) offer is very generous and covers a wide scale of cost and performance.Managing the data using Object Relational DataBase Management System (ORDBMS) 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.

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

1. INTRODUCTION

ORDBMS 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. ORDBMS provides robust data protection and data recovery features. Committed transactions won't be lost, even if your application crashes at an "inopportune" time. Furthermore, ORDBMS 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.

1.1 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 RELATIOANAL DATABASES

A relation is a table of columns and rows. The relation (also called a table) is a finite subset of the Cartesian product of a set of domains, each of which is a set of values Each attribute of the relation (also called a column) corresponds to a domain (the type of the column). The relation is thus a set of tuples (also called rows). Organization Address table, which must appear in the Organization table. You can also constrain several attributes together, such as a primary key consisting of several attributes (Address ID and Organization Name, for example) or a conditional constraint between two or more attributes. You can even express relationships between rows as logical constraints, though most RDBMS products and SQL do not have any way to do this. Another term you often hear for all these types of constraints is "business rules," presumably on the strength of the constraints' ability to express the policies and underlying workings of a business

2.2 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.

2.3 MAPPING RULES

Pure Relational Mapping

There are several commercial POSs mapping OO structures to relational tables. Objects are represented by table rows. Since RDBMSs do not support set-valued attributes, user-defined data types, and object references, additional tables are required to store corresponding data and to connect them with the corresponding class tables . Thus, several tables, may be required to map a given class. Principally, there are several ways of representing a class hierarchy in the relational model.

2.4 KEY CONSIDERATION

1. Full Relational Database Functionality
2. Lower Price & Total Cost of Ownership
3. Cross-platform Portability
4. Delivering a Better "Out-of-the Box" Experience
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

ORDBMS 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. 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.

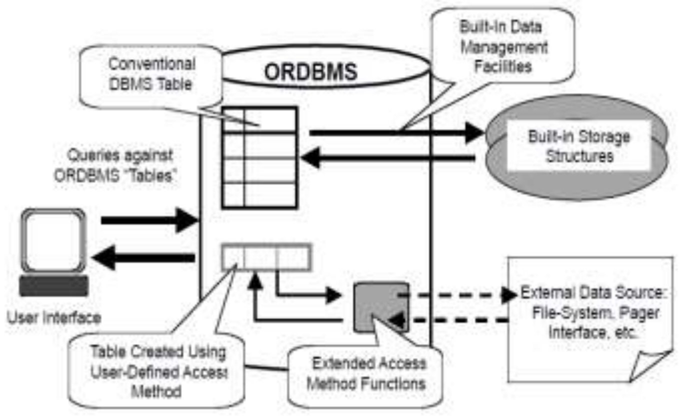

ORDBMS
Conventional DBMS Table
Built-In Data Management Facilities
Built-in Storage Structures
Queries against ORDBMS "Tables"
User Interface
External Data Source: File-System, Pager Interface, etc.
Table Created Using User-Defined Access Method
Extended Access Method Functions

Fig -3.1 Extensible Storage Management

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:

```
Class.forName("com.quadcap.jdbc.JdbcDriver");
java.sql.Connection conn = java.sql.DriverManager.getConnection("jdbc:qed:mydb");
```

3.3 DESIGN OF THE OBJECT-RELATIONAL

Relational Database

RDBMS products worked acceptably in the kinds of high-end, online transactions processing applications served so well by earlier technologies. Despite the technical shortcomings RDBMS technology exploded in popularity because even the earliest products made it cheaper, quicker, and easier to build information systems. For an increasing number of applications, economics favored spending more on hardware and less on people. RDBMS technology made it possible to develop information systems that, while desirable from a business management point of view, had been deemed too expensive

Application Programming Interfaces

There is always more to an information system than a database. Other, external programs are responsible for communication and for managing user interfaces. An information system will use a variety of languages to implement the non-DBMS parts of the final system. Consequently, there are several mechanisms for handling applications programming interfaces (API).

Object -Relational Database

ORDBMS predict their four-quadrant view of the database world, ORDBMS has been the most appropriate DBMS that processes complex data and complex queries. The object-oriented database management systems have made limited inroads during the 1990's, but have since dying off. Instead of a migration from relational to object-oriented systems, as was widely predicted around 1990, the vendors of relational systems have incorporated many object-oriented database features into their DBMS products. As a result, many DBMS products that used to be called "relational" are now called "object-relational".

3.4 THE BACKGROUND OF ORDBMS

The success of Relational DBMSs in the past decades is evident. However, the basic relational model and earlier version of SQL proved inadequate to support object presentation. It has been said that traditional SQL DBMSs experience difficulty when confronted with the kinds of "complex data" found in application areas such as hardware and software design, science and medicine, document processing, mechanical and electrical engineering, etc. To meet the above challenges, the object-relational DBMS emerged as a way of enhancing the capabilities of relational DBMS with some of the features that appeared in object oriented DBMSs. Object-relational DBMSs are supposed to combine the traditional benefits of relational systems with the ability to deal with complex data—a kind of "one size fits all" solution to the database management problem. The concept for the ORDBMs is a hybrid of the RDBMs and OODBMs. The ORDBMS provides a natural and productive way to maintain a consistent structure in the database. Query and procedural languages and call interfaces in ORDBMSs are familiar: SQL3, vendor procedural languages, and ODBC, JDBC, and proprietary call interfaces are all extensions of RDBMS languages and interfaces. As an evolutionary technology.Tthe object relational DBMS has inherited the robust transaction- and performance-management features of its relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and data definition languages (DDLs) while assimilating new object-management possibilities.

3.5 DEVELOPMENT FLOW OF THE APPLICATIONS WITH ORDBMS

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. Analyzing the top-down and bottom-up strategies for implementing the informatics systems, we conclude that, if using object relational databases,

it is appropriate to choose any of them based on the existing system characteristics.A top-down approach is appropriate and desirable in a situation where there is not implemented an informatics system or if there are disparate applications which work as a result of local developments made over time, but their preservation is not vital.

We consider that it is appropriate to achieve an unified and homogeneous system using top-down development strategy, considering that integrating existing databases it is very possible that they are not built using an object relational data model and they use different DBMS. Based on standard steps to be taken into consideration in order to develop database applications, we believe that the sequence of activities must be that shown in the Figure.

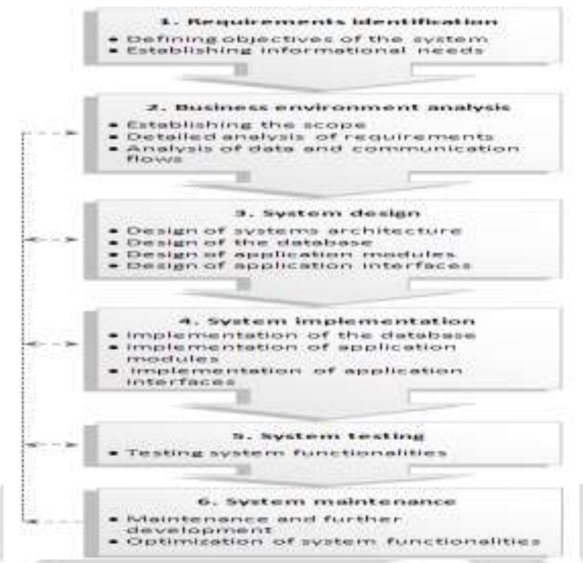

1. Requirements identification
• Defining objectives of the system
• Establishing informational needs
2. Business environment analysis
• Establishing the scope
• Detailed analysis of requirements
• Analysis of data and communication flows
3. System design
• Design of systems architecture
• Design of 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

Fig 3.5: Development flow of the applications with object-relational databases

3.6 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 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..

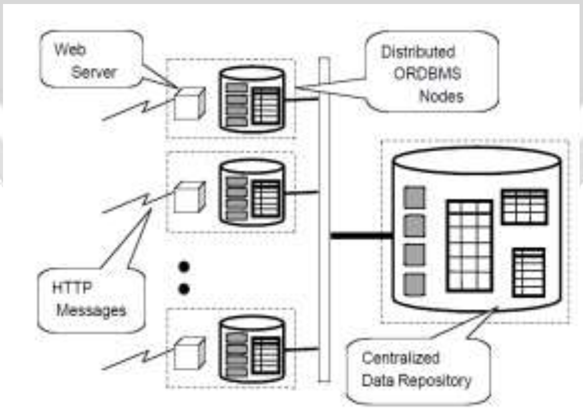
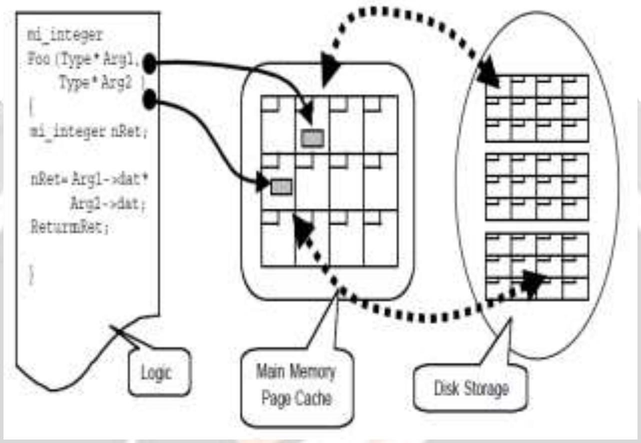


Fig 3.6: Distributed Information System Deployment

3.7 Systems Architecture Possibilities

Having implemented several such functions, the next problem confronting the development team was that although their system worked fine, it would be more useful to include data that was stored in another database. In fact, because their operations were international, they needed to transfer data across significant distances and convert data along the way. Bulk copy and load was considered unfeasible, because the remote data was volatile, and although intersite queries were rare, accurate answers were critical.None of the other groups wanted to migrate off their efficient and stable production systems. The basic problem was that the data lived "out there" in flat files, in another information system, or in another DBMS. Nonetheless, the local business users still wanted to access it. The developers decided to use the external data access features of the ORDBMS to turn it into a federated database system.



```
mi_integer
Foo (Type*Arg1,
     Type*Arg2 )
{
mi_integer nRet;

nRet= Arg1->dat*
      Arg2->dat;
ReturnnRet;

}
```

Fig 3.7: Overview of ORDBMS Table DataBase Management System

3.8 Data Storage in an ORDBMS

Data storage in an ORDBMS is more or less unchanged from what it was in the RDBMS. Data is organized into pages. A set of pages (possibly many) holds all data for a particular table. Blocks of memory are reserved by the ORDBMS for caching frequently accessed pages to avoid the cost of going to disk each time the page is touched.

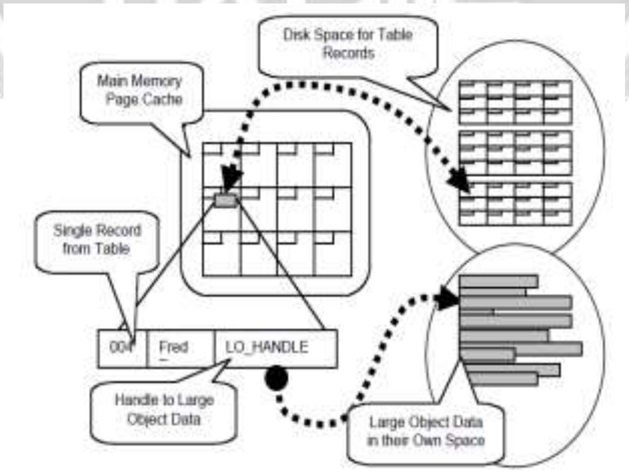

Disk Space for Table Records
Main Memory Page Cache
Single Record from Table
004
Fred
LO_HANDLE
Handle to Large Object Data
Large Object Data in their Own Space

Fig 3.8: ORDBMS Management of Large Object Data

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.

5.SUMMARY AND CONCLUSIONS

5.1 SUMMARY

In this chapter we have introduced object-relational DBMS technology, illustrated something of the role ORDBMSs will play in developing information systems to support the modern enterprise, and briefly described the major components of the technology. An ORDBMS • is a data repository that can be extended to manage any kind of data and to organize any kind of data processing, • represents the next logical step in the evolution of DBMS technology. It both preserves those aspects of relational DBMS technology that proved so successful (abstraction, parallelism, and transaction management) and augments the features with ideas, such as extensibility and advanced data modeling, derived from object-oriented approaches to software engineering, can be thought of as a software back plane, which is a general framework within which new pieces can be integrated as occasion demands. 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

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.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.

6. REFERENCES

1. Begg, C., & Connolly, T. (2010). Database systems: A practical approach to design, implementation, and management, 5th Ed. Addison Wesley.
2. Cho, W., Hong, K. & Loh, W. (2007). Estimating nested selectivity in object-oriented and object-relational databases Information and Software Technology, (49)7, 806-816
3. Connolly, T. and Begg, C. (2006). Database systems: A practical approach to design, implementation, and management, 4th Ed. Addison Wesley.
4. Elmasri, R. & Navathe, S. (2011). Fundamentals of Database Systems, 6th Edition, Addison Wesley.
5. Garcia-Molina, H., Ullman, J. & Widom, J. 2003. Database Systems: The Complete Book, Prentice Hall, Upper Saddle River.
6. He, Z., & Jérôme, D. (2005). Evaluating the Dynamic Behavior of Database Applications, Journal of Database Management; 16:2, 21-45.
7. Hoffer, J., Prescott, M., & Topi, H., 2009 Modern Database Management, 9th Edition, Pearson Prentice Hall.
8. Krishnamurthy, Banerjee and Nori, 1999. Bringing object-relational technology to the mainstream, Proceedings of the ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems, Philadelphia, PA