International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 3, March 2013 ISSN 2319 - 4847

Applying Object Oriented Concepts to RDBMS

Anusha Paruchuri[1], Ch. Phani Krishna[2], V.Samson Deva Kumar[3]

[1]Student, IV/IV B.Tech, Department of Computer Science and Engineering, K L University, Vaddeswaram, Andhra Pradesh
[2]Associate Professor at K L University, Department of Computer Science and Engineering Vaddeswaram, Andhra Pradesh

[3]Project Manager in South Central Railway WWO, S/W Training and Development center, E-world, Vijayawada, A.P

ABSTRACT

Today,most of theClient-Server applications rely on database as a data store for servicing requests from multiple clients.Data organization and management have become so complex and challenging in this electronic age of information.The database technologies have constantly evolved to meet these changing requirements by adopting object oriented programming concepts. Most of the applications use a Relational Database Management System (RDBMS) as their data store. The main theme of this paper is incorporating object-oriented programming concepts into existing relational databases. Object oriented programming concepts such as encapsulation, polymorphism and inheritance are enforced as well as database management concepts such as the ACID properties (Atomicity, Consistency, Isolation and Durability) which lead to the efficient integration. Mainly concepts like inheritance and polymorphism are employed.Nowadays, the necessity to support complex data in databases is intensified. Our main objective is to reduce the implementation overhead,complexity and the memory space required for storage when compared to the traditional databases.The object-oriented development paradigm has the advantage of being a more natural way to model the "real world" of the application domain.

Keywords: Object Oriented Programming concepts, RDBS-relational database systems, ORDBMS (object- relational DBMS),inheritance,polymorphism,Encapsulation.

1. INTRODUCTION

For many years, Relational database[RDB] has been the only solution forefficient storage and retrieval of huge volumes of data The RDBs are based on tables whichare static components of organizationalinformation. In addition to this, RDB can handleonly simple predefined data types.The types of data that can be stored in a table are integer, string, and decimal.The main drawback of RDBs is it cannot handle complex data types,user defined data types and multimedia.Thus, the RDB technology fails in this aspect of handling complex information systems. Often many relationships in RDBs cannot be extracted without user's and are left unexplored. The success of Relational DBMSs in the past decades isevident[3]. However, the basic relational model and earlierversion of SQL proved inadequate to support objectpresentation. It has been said that traditional SQL DBMS experience difficulty when confronted with the kinds of"complex data" found in application areas such as hardwareand software design, science and medicine, documentprocessing, mechanical and electrical engineering, etc.To meet the above challenges, the object-relational DBMSemerged as a way of enhancing the capabilities of relationalDBMS with some of the features that appeared in objectorientedDBMSs. Object-relational DBMSs are supposedto combine the traditional benefits of relational systemswith the ability to deal with complex data—a kind of "onesize fits all" solution to the database management problem .The evolution of ORDBs (object-relational database) thus, came into limelightin order to solve these problems of RDBMS. The traditional RDBMS extended to include Object Oriented concepts and structures such as abstract datatype, nested tables and varying arrays. Based on the concept of abstractionand generalization, object oriented modelscapture the semantics and complexity of the data.. Therefore, many organizations areemploying object-oriented concepts to RDBs for solving their problems of data storage, retrieval and processing. This can be called or shortened as ORDBS (object- relational DBMS).The principal strength of ORDB is itsability to handle applications involving complexand interrelated information. ORDBS was mainly created to handle new types of data such as audio, video, and image files that relational databases were not equipped to handle. There are numerous applicationsbuilt on existing relational database managementsystems (RDBMS), so it's difficult, to eliminate those RDBs. Hence, weintend to include the object-orientedconcepts into the existing RDBMSs, therebymanipulating the features of RDBMSs and OOconcepts.

2. OBJECT-RELATIONAL DATABASE (ORDB)

The chief advantage of ORDB is itsability to represent real world concepts as datamodels in an effective and presentable manner. This system simply puts an object oriented front end on a relational database (RDBMS). When applications interface to this type of database, it will normally interface as though the data is stored as objects. However thesystem will convert the object information into data tables with rows and columns and handle the data the same as a relational database. Likewise, when the data is retrieved, it must be reassembled from simple data into complex objects.The main benefit to this type of database lies in the fact that the software to convert the object data between a RDBMS format and object database format is provided. Therefore it is not necessary for programmers to write code to convert between the two formats and database access is easy from an object oriented computer language.An ORDBs is a database management system with that is similar to a relational database, except that it has an object-oriented database model. This system supports objects, classes and inheritance in database schemas and query language. ORDBMS was created to handle new types of data such as audio, video, and image files that relational databases were not equipped to handle. In addition, its development was the result of increased usage of object-oriented programming language;ORDBs provide a middle ground between relational and object-oriented databases. In ORDBs, data is manipulated using queries in a query language. These systems bridge the gap between conceptual data modeling techniques such as entity relationship diagrams and objectrelational mapping using classes and inheritance. ORDBs also support data model extensions with custom data types and methods. This allows developers to raise the abstraction levels at which problem domains are viewed. ORDBMS may be the most appropriate choice for a database management system (DBMS) that processes complex data and complex queries according to StoneBraker's four-quadrant view of the database world [1]

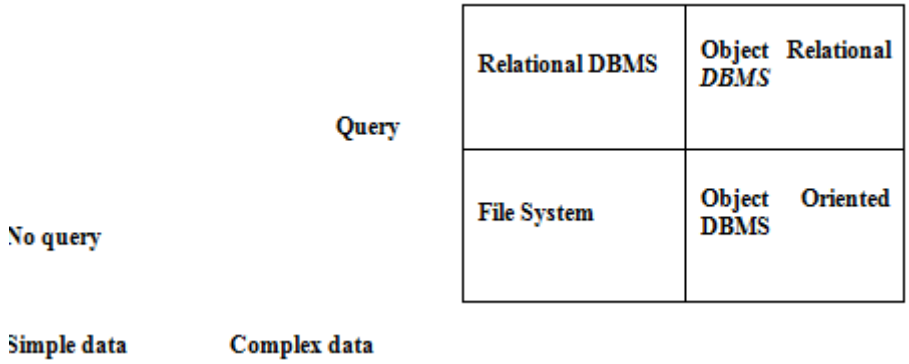

Figure 1: DBMS overview

Lower Left Quadrant-Those application that process simple data andrequire no query capability e.g. text processors.Upper Left Quadrant-Those applications that process simple data andrequire complex query capability.Lower Right Quadrant-Those application that process complex data andrequire no query capability.Upper Right Quadrant-Those applications that process complex data andrequire complex query capability.ORDBs allow developers to integrate the database with their own custom data types and methods. Whereas RDBMS or SQL-DBMS products focused on the efficient management of data drawn from a limited set of data types (defined by the relevant language standards), ORDBs allows software developers to integrate their own types and the methods that apply to them into the DBMS. The goal of ORDBMS technology is to allow developers to raise the level of abstraction at which they view the problem domain. ORDBs support objects, classes and inheritance in database schemas and in the query language. In addition, it supports extension of the data model with custom data-types and methods. It provides a middle ground between relational databases and object-oriented databases (OODBS). In object-relational databases, the approach is essentially that of relational databases: the data resides in the database and is manipulated collectively with queries in a query language; at the other extreme are OODBs in which the database is essentially a persistent object store for software written in an object-oriented programming language, with a programming API for storing and retrieving objects, and little or no specific support for querying. An ORDBS supports an extended form of SQL called SQL3 that is still in the development stages.. 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). Object Relational DBMSs involve the extension of relational database systems to add object-oriented features or direct representation of application objects in relational databases. Object-Relational Database Systems (ORDBs) allow users to define datatypes, functions and operators. As a result, the functionality of the ORDBs increases along with their performance.

An example schema of a student relation which ORDBMS supports

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

In this example we have extra attributes "location" and "picture" which are not present in the traditional EMPLOYEE relation of RDBMS. The datatype of "location" is "geographic point" and "picture" is "image".

ORDBs attempt to extend relational database systems with the functionality necessary to support a broader class of applications and, in many ways, provide a bridge between the relational and object-oriented paradigms. Advantage of ORDBS is that it allows organizations to continue using their existing systems, without having to make major changes and it also allows users and programmers to start using object-oriented systems in parallel. When developing ORDBs, we can include methods and data types. This increases ability to sort through and locate files within these databases at a faster rate. By assigning these data types to files, we can better filter them through the database and also helps in retrieving files that share same characteristics.

3. OVERVIEW OF OOP CONCEPTS

Existing object-oriented systems exhibit significant differencesin their support of the object-oriented paradigm. In this section, to establishour terminology, we review the basic object concepts which we have selected for our data model from existing object oriented programming languages and systems Objects, Attributes (Instance Variables), Methods, and Messages [6].In object-oriented systems, all conceptual entities are modeled as objects. An ordinary integer or string is as much as an object as is a complex assembly of parts, such as an aircraft or a submarine. An object consists of some private memory that holds its state. The private memory is made up of the values for a collection of attributes. The value of an attribute is itself an object, and therefore has its own private memory for its state (i.e., its attributes). A primitive object, such as an integer or a string, has no attributes. It only has a value, which is the object itself. More complex objects contain attributes, through which they reference other objects, which in turn contain attributes. The behavior of an object is encapsulated in methods. Methods consist of code that manipulate or return the state of an object. Methods are a part of the definition of the object. However, methods, as well as attributes, are not visible from outside of the object. Objects can communicate with one another through messages. Messages constitute the public interface of an object. For each message understood by an object, there is a corresponding method that executes the message. An object reacts to a message by executingthe corresponding method, and returning an object.

3.1Classes

If every object is to carry its own attribute names and its own methods, the amount of information to be specified and stored can become unmanageably large. For this reason, as well as for conceptual simplicity, ‘similar’ objects are grouped together into a c/ass. All objects belonging to the same class are described by the same attributes and the same methods. They all respond to the same messages. Objects that belong to a class are called instances of that class. A class describes the form (attributes) of its instances, and the operations (methods) applicable to its instances. Thus, when a message is sent to an instance, the method which implements that message is found in the definition of the class. Classes are used to define a set of similar objects. Objects having same attributes and respond to same messages can be grouped together to form a class.

3.2Class Hierarchy and Inheritance

Grouping objects into classes helps avoid the specification and storage of much redundant information. The concept of a class hierarchy further reduces information redundancy. A class hierarchy is a hierarchy of classes in which an edge between a pair of nodes represents the IS-A relationship; that is, the lower level node is a specialization of the higher level node (and conversely, the higher level node is a generalization of the lower level node). For a pair of classes on a class hierarchy, the higher level class is called a superclass, and the lower level class a subclass. The attributes and methods (collectively called properties) specified for a class are inherited (shared) by all its subclasses. Additional properties may be specified for each of the subclasses. A class inherits properties only from its immediate superclass.Since the latter inherits properties from its own superclass, it follows by induction that a class inherits properties from every class in its superclass chain.

3.3Domains of Attributes

In object-oriented systems, the domain (which corresponds to data type in conventional programming languages) of an attribute is a class. The domain of an attribute of a class C may be explicitly bound to a specific class D. Then instances of the class C may take on as values for the attribute instances of the class D as well as instances of sub classes of D.

3.4Class Lattice, Multiple Inheritance, and Name-Conflict Resolution

In many object-oriented systems (including ORION), a class can have more than one superclass, generalizing the class hierarchy to a lattice (directed acyclic graph). In a class lattice, a class inherits properties from each of its super classes. This feature is often referred to as multiple inheritances.The class lattice simplifies data modeling and often requires fewer classes to be specified than with a class hierarchy. However, it gives rise to conflicts in the names of attributes and methods. One type of conflict is between a class and its superclass (this type of problem also arises in a class hierarchy). Another is among the super classes of a class; this is purely a consequence of multiple inheritance Name conflicts between a class and its super classes are resolved in all systems we are aware of, and in ORION,

3.5 Extensibility

Object Relational Database capabilities are extended with the addition of new data types, access methods and functions found in object-oriented programming. You can add string characters with alpha-numeric data types. Complex data types can combine characteristics of data types that already exist in your database. You can specify data types by the text you wish to contain or by the amount of bytes used to create it. User-defined data types can be opaque or distinctive. You can also add user-defined virtual processors.

3.6 Inheritance

Unlike Relational Databases, Object Relational Databases allow the use of inheritance. Within inheritance, you can develop classes for your data types. These objects can inherit certain capabilities from other classes to be used in other functions of the database. These inherited properties could be something simple like all files within "Class A" are insurance claims. So if you program "Class B" to inherit "Class A" characteristics, then "Class B" will also be insurance claims, but they could also be insurance claims that have been processed recently. Through inheritance, your data types will inherit these features of other data types.Inheritance allows one class to be defined as a special case of a more general class.These special cases are known as subclasses and the more general cases are known as superclasses.

3.7 Polymorphism

Polymorphism in Object Relational Databases involves allowing one operator to have different meanings within the same database. You can connect your tables within your database by building relationships. This includes records that may all contain the same name but different information. Such as if you had records for Joe Doe, but some were insurance claims and some were accident reports. You can connect tables by the name operator; when you query your database to pull records, it will pull all the records containing Joe Doe.

3.8 Encapsulation

You would use encapsulation with Object Relational databases in the form of tables. Say, for instance, you want Table 1 to include name, address, telephone number and email address for your contacts. Through encapsulating the "Contacts" class, you combine all of this information into this one table. So that when you query the database for this information, you generate a report in the style of a form to include all of this information.

4. DEFECTS OF RDBMS

- Difficulty Handling Recursive Queries-Extremely difficult to produce recursive queries.Extension proposed to relational algebra to handle this type of query is unary transitive (recursive) closure operation.

- Impedance Mismatch-Most Data Manipulation Languages lack computational completeness. To overcome this, SQL can be embedded in a high-level 3GL.This produces an impedance mismatch - mixing different programming paradigms. 30% of programming effort and code space is expended on this type of conversion.

- Other Problems with RDBMSs-Transactions are generally short-lived and concurrency control protocols not suited for long-lived transactions.Schema changes are difficult.

- Until recently, RDBs failed to support complex objects such as documents, video, images, spatial or time-series data.

- There is no efficient and effective integrated support for things like text searching within fields.

- Storing and representing some fairly common data structures can be very difficult. Consider a bus route – a simple, ordered list of bus stops. Relational databases only hold tables as unordered lists and can retrieve an ordered list only if a specially built index is added.

- Despite the maturity of relational database products and the dramatic growth in computer power over the past decade, we still hear about projects that fail because the performance of the relational database used is just not good enough. Usually this is because of the way relational databases physically store data. For developers to assemble the data that they need, they often have to do multiple JOINs of one table to another to another to another. To retrieve the data, the database runs optimization routines to determine the best way to gather the data and then retrieves it. This process often takes a long time and can negatively impact performance.

5. DESIGN AND IMPLEMENTATION TOOLS

5.1. Unified Modeling Language (UML*)*

UML is used as a tool for ORDBMS design. UML is a newmodeling tool developed by the Object ManagementGroup. UML development was spearheaded by RationalSoftware Corp. Although the UML technology wasdeveloped mainly for software design, the important part ofthis technology, classes and methods, are roughlyequivalent to ORDBMS types and methods[5].

5.2. Oracle Database Server

The Oracle 8i database server fully supports the Object- relational database model by providing abstract data types, nested tables, varying arrays, binary large objects (BLOB) and character large objects (CLOB) [6]. This provides higher levels of abstraction so that application developer can manipulate persistent objects as opposed to constructing the data from relational data. Moreover, object type declarations can be reused via inheritance, thereby reducing application development time and effort. Three tools inside Oracle 8i were utilized in the author's class as follows:

1. SQLPlus was used to define collection data type.
2. PL/SQL was usedthe implemented user-defined objects and method.
3. JDBC connectivity was used for application programming.

6. FEATURES OF ORDBs

In some cases, object databases have replaced relational databases for performance reasons. This has even been the case in large scale business applications that did not involve the storing of complex objects.

- The major performance advantage that object databases have is that they don't usually have to assemble the data before it can be used the way relational databases do. They tend to store data in its most used form, which typically helps performance.

- Object databases can implement caching strategies that make it more likely that data is in memory when it is requested. They require little optimization to retrieve data.

- The main advantages of extending the relational data model comefrom reuse and sharing.Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.If we can embed the functionality in the server, it saves having to define it in each application that needs it, and consequently allows the functionality to be shared by all applications

- It resolves many of known weaknesses ofRDBMS.Preserves significant body of knowledge and experience gone into developing relationalapplications.

- They run much faster for transactional applications .They handle complex objects far more effectively .They offer better developer productivity .They are easier to manage

7. DRAWBACKS OF ORDBs

- The ORDBMSs approach has the obvious disadvantage of complexity and associated increased costs.There are proponents of the relational approach that believe the essential simplicity and purity of the relational model are lost with these types of extension.
- There are few who believe that the RDBs is being extended for what will be a minority of applications that do not achieve optimal performance with current relational technology.
- When developing a Web application using a relational data base, the data models in the data base and the application are incompatible. This requires a conversion of data from relational to object every time the application accesses data
- The underlying architecture of the OR model renders it wholly inappropriate for developing high-speed Web applications. With an object data base, the same data model exists in both the data base and the application.
- Complexity and increased costs. Supporters of relational approach believe simplicity and purity of relational model are lost.

8. CONCLUSION

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.

REFERENCES

[1] Stonebraker M., Object-relational DBMSs: the Next Great Wave. San Francisco, CA: Morgan KaufmannPublishers, Inc.1996.
[2] AjitaSatheesh “Use of object-oriented concepts in database for effective mining”Vol.1 (3), 2009, 1UIT, RGPV, Bhopal, MP,India.
[3]Fundamental of database Systems, 3rd Ed. by Elmasriand Navathe, Addision-Wesley 2000.
[4] Gheorghe SABĂU “Comparison of RDBMS, OODBMS and ORDBMS”Revista InformaticaEconomică,(44)/2007.
[5]Ming Wang“Implementation of Object-Relational DBMSsin a Relational Database Course”.
[6] Darrell Woelk “Multimedia Information Managementin an Object-Oriented Database System”Proceedings of the 13th VLDB Conference, Brighton 1987.
[7]Prof.S.N. Sawalkar”Implementation of query optimization in OODBMS-Review paper”,International Journal of Modern Engineering Research (IJMER),Vol.1, Issue.2.