Comparative Study of ODBMS vs RDBMS

Brahma Datta Shukla¹, Dr. S. P. Singh²

¹P.hd Research Scholar, Department of Computer Science, NIMS, Jaipur, Rajasthan, INDIA
²HOD & Associate Professor, Computer Science Engineering, NIMS, Jaipur, Rajasthan, INDIA, srigj@rediff mail.com

Abstract

In today environment it is very complicated to choose between the object oriented Model and the relational model, many factors should be considered. The most important of these factors are single level and multilevel access controls, protection against inference, and maintenance of integrity etc. When determining which distributed database model will be more secure for a particular application, the decision should not be made purely on the basis of available security features. One should also question the efficiency of the delivery of these features. Do the features provided by the database model provide adequate security for the Intended application? Does the implementation of the security controls add an unacceptable amount of Computational overhead? In this paper, we describe the object databases advantages and disadvantages over RDBMS. We also give the description of implementation issues of object oriented databases.

Keywords: Data Model, OODBMS, RDBMS, Distributed Database Model etc.

1. INTRODUCTION

As distributed system become more popular, the need for improvement in distributed database management systems becomes even more important. A distributed system varies from a centralized system in one key respect: The data and often the control of the data are spread out over two or more geographically separate sites. Distributed database management systems are subject to many security threats additional to those present in a centralized database management system (DBMS). Furthermore, the development of adequate distributed database security has been complicated by the relatively recent introduction of the object-oriented database model.

For the last several years the most used database model has been relational. While the relational model has been useful, its utility is reduced if the data does not fit into a relational table. Many organizations have data requirements that are more complex than can be handled with these data types. Multimedia data, graphics, and photographs are examples of these complex data types.

1.1 INTRODUCTION TO RELATIONAL MODEL

Relational databases typically treat complex data types as BLOBs (binary large objects). For many users, it is not feasible because BLOBs cannot be queried. In addition, database developers have impedance mismatch between the third generation language (3GL) and structured query language (SQL). The impedance mismatch occurs when the 3GL command set conflicts with SQL. There are two types of impedance mismatches:

(1) Data type inconsistency: A data type recognized by the relational database is not recognized by the 3GL. For example, most 3GLs don’t have a data type for dates. In order to process date fields, the 3GL must convert the date into a string or a Julian date. This conversion adds extra processing overhead.

(2) Data manipulation inconsistency: Most procedural languages read only one record at a time, while SQL reads records a set at a time. This problem is typically overcome by embedding SQL commands in the 3GL code.

Solutions to both impedance problems add complexity and overhead. Object-oriented databases have been developed in response to the problems listed above: They can fully integrate complex data types, and their use eliminates the impedance mismatch.

The development of relational database security procedures and standards is a more mature field than for the object-oriented model. This is principally due to the fact that object-oriented databases are relatively new. The relative immaturity of the object-oriented model is particularly evident in distributed applications.

1.2 INTRODUCTION TO OBJECT ORIENTED MODEL

Complex objects, object identity. Objects have arbitrary complexity. Each object has identity, i.e. a unique internal identifier (OID) (with no meaning in the problem domain), and each object has one or more external names.

Relationships, associations, link. Objects are connected by conceptual links. For instance, the Employee and Department objects can be connected by a link works for Encapsulation and information hiding. The internal properties of an object are subdivided into two parts: public and private (invisible from the outside).

Classes, types, interface. Each object is an instance of one or more classes. Objects are instantiated according to
information presented in the class. The class contains the properties that are common for some collection of objects (objects’ invariants). Each object has a type. Objects are accessible via their interfaces.

Abstract data types (ADTs): a kind of a class, which assumes that any access to an object is limited to the predefined collection of operations. Operations, methods and messages. An object is associated with a set of operations (methods). The object performs the operation after receiving a message with the name of operation to be performed (and its parameters).

Inheritance & Classes are organized in a hierarchy reflecting the hierarchy of real world concepts. For instance, the class Person is a superclass of the classes Employee and Student.

Polymorphism, late binding & overriding. The operation to be executed on an object is chosen dynamically, after the object receives the message with the operation name.

Persistence-Database objects are persistent, i.e., they live as long as necessary. They can outlive, which created them.

### 2. RELATIONAL VS OBJECT DATABASE

<table>
<thead>
<tr>
<th>OODBMS</th>
<th>ORDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is created on the basis of persistent programming paradigm.</td>
<td>It is built by creating object-oriented extensions of a relational database system.</td>
</tr>
<tr>
<td>It supports ODL and OQL for defining and manipulating complex data types.</td>
<td>It supports an extended form of SQL.</td>
</tr>
<tr>
<td>It aims to achieve seamless integration with object-oriented programming languages such as C++, Java, or Smalltalk.</td>
<td>Such integration is not required as SQL: 1999 allows us to embed SQL commands in a host language.</td>
</tr>
<tr>
<td>Query optimization is difficult to achieve in these databases.</td>
<td>The relational model has a very strong foundation for query optimization, which helps in reducing the time taken to execute a query.</td>
</tr>
<tr>
<td>The query facilities of OQL are not supported efficiently in most OODBMS.</td>
<td>The query facilities are the main focus of ORDBMS. The querying in these databases is as simple as in relational database system, even for complex data types and multimedia data.</td>
</tr>
<tr>
<td>It is based on object-oriented programming languages; any error of data type made by programmer may affect many users.</td>
<td>It provides good protection against programming errors.</td>
</tr>
</tbody>
</table>

### 3. OBJECT DATABASE ADVANTAGES OVER RDBMS

These are main advantages of object database over RDBMS is following:
- Objects don't require assembly and disassembly saving coding time and execution time to assemble or disassemble objects.
- Reduced paging
- Easier navigation
- Better concurrency control - A hierarchy of objects may be locked.
- Data model is based on the real world.
- Works well for distributed architectures.
- Less code required when applications are object oriented.

4. OBJECT DATABASE DISADVANTAGES COMPARED TO RDBMS
These are main disadvantages of object database over RDBMS is following:

- Lower efficiency when data is simple and relationships are simple.
- Relational tables are simpler.
- Late binding may slow access speed.
- More user tools exist for RDBMS.
- Standards for RDBMS are more stable.
- Support for RDBMS is more certain and change is less likely to be required.

5. OBJECT ORIENTED DATABASE IMPLEMENTATION ISSUES
There are following issues are used to implement object oriented database.

- Persistence
- Sharing
- Paging
- Back to the Beginning

Object-oriented databases give objects persistence, which enables objects to be stored between database runs. In Object Oriented, paging is one of the memory-management schemes by which a object oriented database can store and retrieve data from secondary storage for use in main memory.

In the paging memory-management scheme, the operating system retrieves data from secondary storage in same-size blocks called pages.

The main advantage of paging over memory segmentation is that it allows the physical address space of a process to be noncontiguous. Before the time paging was used, systems had to fit whole programs into storage contiguously, which caused various storage and fragmentation problems.

There are following advantages of paging in object oriented database-

- Determine the location of the data in auxiliary storage.
- Load the requested data into the available page frame.
- Update the page table to show the new data.

- Return control to the program, transparently retrying the instruction that caused the page fault.
- Easy to maintain

6. CONCLUSIONS
In this paper, we provide the detail characteristics of database by which it is used means mandatory, optional & open. We provide the Object Database Advantages over RDBMS, Object Database Disadvantages compared to RDBMS, Object Oriented Database Standards, How object oriented database is implemented. We also provide how do ODBMS & RDBMS Work & Implementation Issues, benefits & drawbacks of Distributed Database.
7. REFERENCES


[10]. Gupta Dhiraj, Shukla B.D. Security based comparison of Relational and Object Oriented Distributed Database Model, Shodh 04, 2010 www.pioneer.journal.in Page 5..


