OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM

Meenakshi Nair
Faculty, Department of Computer Science

Abstract

The hypothesis with reference to Object oriented Database Management System is to provide an insight into development of implementation operations and maintenance of large complex data intensive application such as computer integrated manufacturing which can be simplified through the OODBMS technology. It focuses on data rather than procedures and gives more security to data. It provides three accesses, specifies private, public and protected, and strictly provides security to data in database. It also provides function as well as data so that the data can be manipulated by the given functions.

Introduction

Data Base Management System (DBMS) provide the facility to users to define, create and maintain the database. It gives easy access for retrieval of data and helps in maintaining computerized record keeping system. A few years back industrial experts pushed object-oriented databases as a technology on rise and said it is well suited for the emerging Internet age.

They said object-oriented database-management systems (ODBMSs) would soon become the primary database technology by succeeding relational database-management systems (RDBMSs). The available RDBMSs were not designed to handle the type of multimedia data frequently found on the Internet. It was also observed that the growth of intranets signaled a decline in the use of client-server networks, in which most relational databases were used.

The key note here is what has resulted into the development of different DBMS systems. Let us move forward and understand them.

Data models

Each of the DBMS is implemented on one of the data models. Data models helps to organize and structure data and information. All the data models consist of three basic components, a set of domain and a set of relation, operation on relation, integrity rule. Based upon the structure and organization, data models are of following types:

1. Hierarchical Model
2. Network Model
3. Relational data base model
4. Object oriented data model

Hierarchical Model

- It is one of the oldest model being used for representation of the data.
- It is similar to the network model in the sense that data and relationship among data are represented by records and links.
- The only difference is that in the Hierarchical model, records are organized as a tree rather than arbitrary graphs.

Drawbacks

- It only supports one to one and one to many relationship.
- Supported data access is navigational.

Network Model

- In Network data model data is represented by a collection of records and relationship among data are represented by links which can be viewed as a pointer.
- A node contains data element and pointer
- The Network data model is similar to a hierarchical model, except that an entity can have many to many relationships.

Drawbacks

- Data Access is navigational
- Needs skilled programmers to manage
- Lacks flexibility to support changing needs

Relational data base model

- The Relational Model is based on predicate logic and set theory. It was first formulated and proposed by Edgar Codd .
- In this model, data is organized in the form of row and column similar to a table. The table is referred to as relations in a relational data model. Rows of the table are referred to as tuples and the columns of a table are referred to as attributes.
- A relational database model is defined as a database which allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.
- Biggest advantage of RDBMS is the ease with which user can create and access data. However, there are few limitations to the relational database management system.

Drawbacks

- Relational database does not have enough storage structure to handle data such as images, digital and audio/video.
- The second limitation of RDBMS is its inadequacy to operate with language outside sql. languages like C++, java, .Net work form. However, RDBMS do not work efficiently with these languages.
- Third limitation is the requirement that information must be in the table from where relationship between entities is defined by values.

OODBMS

OODBMS is a combination of database capabilities and object-oriented programming language capabilities. With the help of OODBMS programmers can develop the product and store them as objects which can be further replicated or modified to make new objects with in OODBMS. Programmer can maintain consistencies within one environment using OODBMS because the database is integrated with the programming language thus model of data representation will be same in database as well as in programming language. In the case of relational DBMS an application project will have clearer division between the database model and the application.

More and more business environments are now switching on to intranet and extranet environments. Companies are showing interest in using OODBMS for representing their complex business data. The DBMS which is specifically designed to store data as objects gives advantage to those companies which are into multimedia presentations or the organizations which uses computer aided designs as a tool.

Characteristics of OODBMS

The most important characteristics of OODBMS is the joining of OODBMS with database technology which provides an integrated application development system. The key element in OODBMS is 'Object'. There are multiple definitions for object, here we will use: *Object is a set of properties and methods for manipulation.* All objects with the same set of properties and methods form a class. Although it is not possible to specify complete set of features which can be used to determine if the particular DBMS is object oriented or not, there are certainly a number of features which are common to existing object-oriented database systems.

Here are few features which are more commonly used and explained precisely:

Object identifier (OID)
In relational database systems records are identified only by their content. If all attributes of two records have the same values, then there is no way to distinguish these two records. But in object oriented databases objects are uniquely identified by OID. Format of OID is specific for each system. In some systems just four bytes with object index or object position in the file is enough to identify the object. In other systems object identifiers are more complex and preserve uniqueness even outside the scope of the local computer.

Object references
References between objects in OO databases are represented using OIDs. So, if field F of persistent object A contains reference to persistent object B, then this field F of object A stores OID of object B. Object oriented database provides efficient ways to access object by it's OID.

Object classes
OODBMS stores objects inside database. In order to make it possible to load/store objects and perform some other operations with them (garbage collection for example), OODBMS should know format of the object, i.e. set of object fields and their types. It is not efficient to store this information for each object instance. Since format of all objects belonging to one class is the same, it is possible to store class and let object reference it's class. Some OODBMS treat classes as normal objects, in which format also has to be defined. In this case the notion of metaclass is introduced. *Metaclass is class of the class.*

Inheritance and polymorphism
All object oriented languages support inheritance. Inheritance is a mechanism allowing child object to inherit behavior and properties of parent object. OODBMS should certainly be able to represent inheritance in database. If class A is derived from class B, then class A inherits all methods of class B and it can be used everywhere where class B can be used. So, it makes it possible to manipulate with instance of class B using variable with type A. It is called polymorphism and its support is one of the main features of object oriented languages and databases.

Tight integration with programming language
Relational databases provide non-procedural query language. It means that query expressed in this language specifies what should be done, but doesn't specify how it is to be done. In contradiction to it, most of the modern programming languages are imperative languages, so the program written in this language specifies at higher or lower level of abstractions which actions should be performed to produce requested result. Object-oriented languages are also imperative languages. Although, OODBMS usually provides non-procedural query language (some kind of object oriented extension of SQL), the main language to access object oriented database is imperative language. So, the main goal of OODBMS is to provide seamless and efficient integration with this language(s).

Traditional DBMS features
Database systems were implemented to provide efficient and consistent way of manipulation with data. So object oriented database systems also need to support these features to be able to be called "database systems". One of the basic notion is ACID (Atomic, Consistent, Isolation, and Durable) transactions. It is not the intention to explain principles of supporting transactions in database systems. The only thing to be noticed is that database should be able to provide concurrent access to the data by many clients and preserve consistency of data.

OODBMS has following advantages and benefits:

OODBMS commonly support certain features that are not available in a RDBMS. These features could be used within application to offer additional value and benefits to application consumers.

- **Versioning:** This feature allows the OODBMS

to manage multiple versions of the same object. In its simplest form, linear versioning makes a sequential series of the same object available to an application. Each subsequent version represents the increment change in the object from the previous version to the next. The idea of a "current" version is also supported. These features would be used by many of the entities in consumer applications. Candidates for versioning include: [batch recipes + process graphics + alarm configurations + block configurations].

- **Schema Evolution:** This feature allows the OODBMS to deal with objects as they evolve from one release of consumer software to the next. Procedures to upgrade an "old" object to a "new" object are handled by the OODBMS.

- **Off-Line Data Bases:** OODBMS offer the concept of online and off-line databases. To the consumer this means a single application may be written to handle both off-line and online configuration. This means that they may run their system with the online data base, but prepare for future system expansions with off-line databases. The configuration work with the off-line database may actually be performed at a different site. The resulting off-line database containing the "new" configuration for the system expansion may be transported to the site via network, portable drives etc. A simple menu selection could merge the off-line database with the consumer's online database.

Long Term Transactions: CAD applications were some of the first users of OODBMS. These applications as well as consumer specific applications require transaction models that are not supported by conventional RDBMS. A user may require long term access to information managed by the database.

As an example, consider a process graphic that may take days to be finished by a configurer. The database must provide exclusive access to the graphic until the configurer is finished and decides to check the graphic back into the database. Only then would the graphic be available to the rest of users. OODBMS provide this type of long term transaction in addition to the conventional locking strategy supported by RDBMS.

Need For OODBMS

The high emphasis on process integration and other business needs are enabling the adoption of object oriented database system. Broadly, we can categorize integration and business needs in following categories:

1. High performance

With complex data, it is not unusual to find that an OODBMS will run anywhere from 10 to 1000 times faster than an RDBMS. The range of this performance advantage depends on the complexity of the data and the access patterns for the data.

OODBMSs are faster because they are optimized for the traversals related to complex data. They also do not have any "impedance mismatch" when it comes to using object programming languages such as Java and C++. High performance can impact business considerations in two ways:

- You simply may need the best performance possible on complex data.
- You may take advantage of the high performance OODBMSs which provide for complex data by purchasing cheaper hardware.

2. Complex data

Complex data is often characterized by:

- A lack of unique, natural identification.
- A large number of many-to-many relationships.
- Access using traversals.
- Frequent use of type codes such as those found in the relational schema

3. One model to reduce development and maintenance cost

Since an OODBMS stores exactly the same object

model that is used at the application level, both development and maintenance costs can be reduced. With an OODBMS, there is no need to:

- Develop two data models: an object model in the application and a relational model stored in the database. This is not needed because an ODBMS uses the same object model as the application.
- Maintain two data models. An OODBMS eliminates the maintenance cost of keeping the data models synchronized.
- Develop mapping between the relational and the object models. This is not needed because an OODBMS uses the same object model as the application.
- Maintain the mapping between the relational and object models. An ODBMS eliminates the maintenance cost of maintaining the mapping whenever there is a change to the object or relational model.

The result is that for a development team of six to seven, it is possible to have a team with one less person when using an OODBMS. It is typical when developing an object application with a RDBMS that one person is in charge of keeping the relational model synchronized with an object model and the mapping code. That person would not be needed in an OODBMS development project.

The following is an example to precisely understand the needs:

Computer integration manufacturing (CIM) area is focusing heavily on the use of OODB technology. The process integration frameworks for advance automation system, hospital patient care tracking system, etc. are using OODB technology. All these applications need to manage complex highly inter-related information. The problem with relational database system is that they require the application developer to force an information model into tables where relationships between entities are defined by value. The relational database is in a process of trying to figure out how to represent real world objects within tables such that good performance results and preservation of data integrity is possible.

These limitations of RDBMS can be overcome with the use of OODBMS as real world entities which can be represented as object. For OODBMS, object design is the fundamental part of overall application designed process.

Suppose we wish to define two objects, classes namely 'Department' and 'Employee' then the necessary class definition for department and employee might somewhat look like:

```
class emp
   { public (EMP# char,
         Ename char,
         Salary money
         position Ref (Job)----

method (----)---;
 }
class dept
 {   public(Dep# char,
        Budget money,
        MGR  Ref (Emp),
        Emps Ref (Emp)____
  method {HIR_Emp [Ref(Emp)]_ _ _code _ _,
           First_Emp [Ref(Emp)]_ _ _code _ _,
}
```

In the above examples, we used access, which specifies the security of data from unauthorized use of data, only class members can access the data. All data and access methods are declared in the class, only defined methods can access the data and manipulate it.

In this way we can provide better security to data and save data from unauthorized access.

Summary and concluding remarks

The OODBMS contains extensive concepts, which makes it a lot more complicated comparing with the RDBMS. This paper gives an introduction of OODBMS concepts.

Based on all this information it is clear that OODBMS provide better information modeling facility than relational DBMS. Following are the advantages:

1. Schema is easier to understand because it is structured, contains more of the semantic of the data and is intuitive.

2. Inheritance allows common attribute to be factorized and stated once. System define object identifies allow compound keys which can potentially become large, to be replaced by a fixed size, smaller and uniform system define object identifiers.

3. Object oriented graphics interface is easier to use than form oriented character base interface. This OOGUI allows user to get to the desire information with fewer actions. [Mouse Click, Menu Selection].

Another important observation that could be made is that an OODBMS can be implemented independent of the application. This is mostly because encapsulation of data and the abstraction of implementation from use through functions.

Reference

- Konstantin Knizhnik : http://www.garret.ru/oodbms.html
- Kim, Won. *Introduction to Object-Oriented Databases*
- W. Kim, *A foundation for object-oriented databases*
- K. R. Dittrich, *Advances in Object-Oriented Database Systems.*