Comparison of RDBMS, OODBMS and ORDBMS

Gheorghe SABĂU, Bucharest, Romania

The presentation of the similarities and differences between relational modeling of data and the object oriented modeling of data is of great importance both for data base designers and for users.
By being well acquainted with the relational model and by noting the similarities and differences between the two approaches to data modeling, designers will be able to turn into account and to make use of the already acquired experience as an important basis for understanding and learning the methodology of designing object oriented databases.
At the time if designers know the similarities and differences between these two approaches they have the possibility to convert a relational model into an object oriented model and inversely.
In our presentation below we will treat RDBMS, OODBMS and ORDBMS comparatively.

Comparing the RDBMS with the OODBMS

1.1. Comparing the RDBMS with the OODBMS as far as data modeling is concerned.

The essential distinction between these two types of data modeling is represented by the encapsulation in the object of both is state and behavior with the object oriented model, while with the relational model only the state is evidenced.
As we all know a relational database is made up of relations, who are sets of tuples, while an object-oriented database is made up of classes, which are sets of classes.
Thus, a relational database will contain a relation called STUDENT, with tuples containing information about each student, while a relational database will contain a class called STUDENT, with object containing information about each student.
It is to be noted that there is the possibility of converting the object model into a relational mode.
In such a situation each class corresponds to a relation, the attributes of a particular class will become attributes corresponding to a relation and the same time, each object instance in a class will have a corresponding tuple in a relation.
While in a relational database the components of a tuple must be primitive types (strings, integer, real, etc.), in an object-oriented database the components of an object may be complex types (sets, tuples, objects, etc.). Table 1 presents a comparison of the main concepts used in object and relational modeling of data.

Table 1. Comparing OODBMS and RDBMS as far as data modeling is concerned.

Object oriented model	Relational Model	Differences
Object	Entity	The object specifies behavior too
Class of objects	Types of Entities	The class of objects includes the common behavior of objects in that class
Class hierarchy	The data base scheme	The class hierarchy includes inheritance, while the scheme includes external keys
Class instance	Entity, tuple or record	The instance may have a more restrictive character
Attribute	Attribute	There are no differences
Relations	Relations	There are no differences They have the meaning of descriptions but with the OODBMS the inheritance includes both the state and the behavior

Messages/Interface	There are none	
Encapsulation	There is none	
Object identifier (OID)	Primary key	In the relational model if the primary key is not identified the system generates an identifier automatically
Inheritance	There is none	

1.2. Comparing RDBMS with OODBMS as far as their targeted objectives are concerned. The difference between the OODBMS and RDBMS may be also put into light by considering their objectives and other characteristics as it can be seen in table 2.

Table 2. Comparing OODBMS with RDBMS considering their objectives

OODBMS	**RDBMS**
• Main objectives: data encapsulation and independence.	• Main objective: ensuring data independence from application programs.
• Independence of classes: classes can be reorganized without affecting the mode of using them.	• Data independence: Data can be reorganized and modified without affecting the mode of using them.
• OODBMS store data and methods.	• RDBMS store only data.
• Encapsulation: the data can be used only through their classes' methods.	• Data partitioning: data can be partitioned depending on the requirements of the users and on the specific users applications.
• Active objects: the objects active. Requests cause objects to execute their methods.	• Passive data: the data are passive. Certain operations, which are limited, can be automatically brought into use when the data are used.
• Complexity: the structure of data may be complex, involving different types of data.	• Simplicity: users perceive data as columns, rows/tuples and tables.
• Chained data: data can be chained so that the methods of classes may bring about increased performance. Structured data such as BLOBS (binary large objects) are used for sound, image, video etc.	• Separate Tables: each relation/table is separate. The Join Operator refers data from separate tables.
• Non-redundancy of methods: data and methods non-redundancy is achieved through encapsulation and inheritance. Inheritance helps to reduce the redundancy of methods.	• Data non-redundancy: data normalization aims at eliminating or reducing data redundancy. It is used in the stage of designing the database and not in the stage of developing the applications.
• Optimizing classes: the data for an object can be interrelated and stored together, so that they may all be accessed by the access mechanism.	• RDBMS performance is related to the level of complexity of the data structure.
• Consistent conceptual model: the models used for analysis, designing, programming and accessing the database are similar. The classes of objects directly represent the concepts of applications.	• Different conceptual model: the model of data structure and data access represented by tables and JOINS is different from the model of analysis, designing and programming. The project must be converted in relational and access tables in accordance with SQL.

2. Comparing the RDBMS with the ORDBMS

When we compare the RDBMS with the ORDBMS the following aspects can be noted:

- An ORDBMS is a relational DBMS with SQL_3 extensions.
- SQL_3 extensions include: row types, user-defined types and user-defined routines, polymorphism, inheritance, reference types and object identity, collection types (ARRAYs), new language constructs that make SQL computationally complete, triggers and support for language objects – Binary Large Objects (BLOBs) and Character Large Objects (CLOBs) – and recursion.
- A RDBMS is characterized by simplicity and increased stability as compared to an ORDBMS, and this fact confers it the quality of being easily used.
- Traditional RDBMS use B – tree indexes to speed access to scalar data. With the ability to define complex data types in an ORDBMS, specialized index structures are

required for efficient to data. Some ORDBMSs are beginning to support additional index types, such as generic B-trees, R-trees (region trees) for fast access to two – and three dimensional data, and the ability to index on the output of a function.

- A mechanism to plug in any user – defined index structure provides the highest level of flexibility.
- Both DBMSs are characterized by simplicity of development owing to the fact that it provides independence of data from applications good for simple relationships.
- For RDBMS there is SQL_2 standard (ANSI X3H2) and for ORDBMS there is SQL_3 standard.
- RDBMS is a mature software product while ORDBMS is an immature product (extensions are new, thy are still being defined and are relatively unproven.
- As far as the support for object – oriented programming is concerned, with the RDBMS, programmers spend 25% of coding time mapping the program object to the database, and with the ORDBMS, the support for object-oriented programming is limited mostly to new data types.

3. Comparing OODBMS with ORDBMS

When we compared OODBMS with ORDBMS some conclusions can be drawn.

- OODBMSs and ORDBMSs both support user-defined ADTs, structured types, object identity and reference types, and inheritance;
- They both support a query language for manipulating collection types;
- ORDBMSs support an extended form of SQL, and OODBMSs support ODL/OQL;
- ORDBMSs consciously try to add OODBMS features to an RDBMS, and OODBMS in their turn have developed query languages based on relational query languages;
- Both OODBMSs and ORDBMSs provide DBMS functionality such as concurrency control and recovery;
- OODBMSs try to add DBMS functionality to a programming language, whereas ORDBMSs try to add richer data types to a relational DBMS.

Conclusions

From the literature we can draw some conclusions regarding RDBMS and OODBMS:

- Relational databases have as their objective to ensure data independence. Normalized data are separated from processing and the processing corresponding to satisfying informational requirements need not be totally pre-defined, thus accepting ad-hoc requirements too.
- Object oriented databases have as their main objective encapsulation, being stored together with the data and the methods. They are inseparable. It is said that we have to do with an independence of classes and not with an independence of data.
- An OODBMS and not an RDBMS is needed when in the reference applications we have to do with complex data.
- The object oriented database markets will continue to develop, but they will still (represent) only a fraction of the traditional databases.
- It is appreciated that RDMSs hold the largest part of the largest part of the databases. But the prospect is that they will still co-exist for a long time future with the OODBS.

References

1. Atkinson M., Bancilhou F. – Object Oriented Database System Manifest. In Proc. 1st Int. Conf. Deductive and Object Oriented Database, Kyoto, Japan, 1989.
2. R.G.G. Cattell – Object Data Management., Ed. Addison Wesley, 1994.
3. Codd E.F. – Is your DBMS really relational?, Computerworld, oct the 14th 1985.
4. Codd E.F. – Does your DBMS run by tech rules?, Computerworld, 21 oct. 1985.
5. C. J. Date – An Introduction to Database Systems. Eight Edition, Pearson Addison-Wesley, 2004.
6. Thomas Connolly, Carolyn Begg – Database to Design, Implementation and Management. Fourth Edition, Ed. Addison-Wesley, 2005.