Journal of Computer Science 2 (10): 781-784, 2006
ISSN 1549-3636

Migration from Relational Database into Object Oriented Database

[1]Mansaf Alam and [2]Siri Krishan Wasan
[1]Department of Computer Science, Jamia Millia Islamia, New Delhi-25, India
[2]Department of Mathematics, Jamia Millia Islamia, New Delhi-25, India

Abstract: Object – Oriented Technology is an important discipline in the field of software engineering in general and it is, therefore, natural to ask whether it is relevant to the field of database management in particular and what that relevance is. There is, however, no consensus on answer to these questions. Some authorities believe that object oriented database systems will take over the world replacing relational system whereas others believe that they are suited only to certain very specific problems and will never capture more than a small fraction of the overall market. Object-Oriented technology represents real world very well, focusing on data rather than procedure and gives more security to data. Also, it is safe from unauthorized use of data because it provides three access specifiers viz. private, public and protected and strictly provides security to data in database. This technology also provides function as well as data together so that the data can be manipulated by the given function. In this paper, we show that how the data is more secure in object-oriented database than in relational database and also why do we migrate from RDBMS into OODBMS

Key words: Versant ODBMS, object-oriented database

INTRODUCTION

Relational database: A relational database stores all its data inside tables and nothing more. All operations on data are done on the tables themselves or produce another tables as the result. You never see anything except that tables. A table is a set of rows and columns and a set does not have any predefined sort order for its elements. Each row is a set of columns with only one value for each. All rows from the same table have the same set of columns, although some columns may have NULL values, i.e. the value for that row is not initialized. It is to be noted that a NULL value for a string column is different from an empty string. As an example, the Relational model[1,2] supports relations, which are set of tuples with fixed number of primitive data elements. The rows from a relational table are analogous to a record and the columns to a field. Here's an example of a table and the SQL statement that creates the table:

```
  CREATE TABLE ADDR_BOOK (
                    NAME char(30),
                    COMPANY char(20),
                    E_MAIL char (25)  )
+-----------------+--------------+--------------------+
 NAME  | COMPANY      | E_MAIL
|+===========+==========+===========+
| Israr Ahmad| Software System lisrarl@centroin.com.
|
+------------------+--------------+--------------------+
| Abid       | IBM   | Abidl@ibm.com    |
+------------------+--------------+--------------------+
```

There are two basic operations that we can perform on a relational table. Viz. Retrieving a subset of its columns and retrieving a subset of its rows. Here are samples of the two operations:

```
SELECT NAME, E_MAIL FROM ADDR_BOOK
+-------------------+-----------------------+
| NAME              | E_MAIL                |
+===================+=========+
| Israr Ahmad   | israr@centroin.com.br   |
+-------------------+-----------------------+
| Abid       |  abidl@ibm.com   |
+-------------------+-----------------------+
SELECT * FROM ADDR_BOOK WHERE COMPANY =
'Software System'
+-------------------+---------------+---------------------+
| NAME        | COMPANY       | E_MAIL       |
+========+==============+==========+
| Israr  Ahmad  |Software System | israr@centroin.com |
+-------------------+---------------+---------------------+
We  can also combine these two operations as follows:
SELECT NAME, E_MAIL FROM ADDR_BOOK WHERE
COMPANY = 'Software system'
+-------------------+-----------------------+
| NAME              | E_MAIL                |
+============+=============+
| Israr Ahmad   |israr@centroin.com.br   |
+-------------------+-----------------------+
```

We can also perform operations between two tables treating them as sets: we can make Cartesian product of the tables and can get the intersection between two tables, we can add one table to another and so on. Later we should be discussing these operations in OODBMS and show how they are more useful and better.

Corresponding Author: Mansaf Alam, Department of Computer Science, Jamia Millia Islamia, New Delhi-25, India

Object oriented databases: In this paper, we examine object systems by introducing and explaining basic object oriented concepts and offer some opinion regarding the suitability of incorporating such concepts into the database systems of the future. The advent and commercial success of well-engineered ODBMS products, such as ObjectStore[3], indicate that the time is ripe to seriously investigate migration from RDBMS to ODBMS.

The classical SQL systems being inadequate in a variety of ways, we are led to study object systems.

The need for object-oriented databases: The increased emphasis on process integration is a driving force for the adoption of object-oriented database systems. For example, the Computer Integrated Manufacturing (CIM) area is focusing heavily on using object-oriented database technology as the process integration framework. Advanced office automation systems use object-oriented database systems to handle hypermedia data. Hospital patient care tracking systems use object-oriented database technologies for ease of use. All of these applications are characterized by having to manage complex, highly interrelated information, which is the strength of object-oriented database systems. Clearly, relational database technology has failed to handle the needs of complex information systems. The problem with relational database systems is that they require the application developer to force an information model into tables where relationships between entities are defined by values. Mary Loomis, the architect of the Versant OODBMS compares relational and object-oriented databases as follow[4]. Relational database design is really a process of trying to figure out how to represent real-world objects within the confines of tables in such a way that good performance results and preserving data integrity are possible. Object database design is quite different. For the most part, object database design is a fundamental part of the overall application design process. The object classes used by the programming language are the classes used by the ODBMS. Because their models are consistent, there is no need to transform the program's object model to something unique for the database manager[5]. An initial area of focus by several object-oriented database vendors has been the Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Computer Aided Software Engineering (CASE) applications. A primary characteristic of these applications is the need to manage very complex information efficiently. Other areas where object-oriented database technology can be applied include factory and office automation. For example, the manufacture of an aircraft requires the tracking of millions of interdependent parts that may be assembled in different configurations. Object-oriented database systems hold the promise of putting solutions to these complex problems within reach of users.

Object-orientation is yet another step in the quest for expressing solutions to problems in a more natural, easier to understand way. Michael Brodie in his book On Conceptual Modeling[6] states, "The fundamental characteristic of the new level of system description is that it is closer to the human conceptualization of a problem domain". Descriptions at this level can enhance communication between system designers,

Object-oriented concept: The object-oriented paradigm is the latest in the software development and the most adopted one in the developing project of today. RDBMS extensions have been spurred by competition from object-oriented database management systems (ODBMSs), which combine comprehensive database management functionality and full-fledged OO data modeling[7].

Limitation of Procedural Programming: A Program in a procedural language is a list of instructions where each statement tells the computer to do something. The focus is on the processing, the algorithm needed to perform the desired computation.

* In procedural paradigm, the emphasis is on doing things. And not on the data. But Data is, after all, the reason for a program's existence. The important part of an inventory program isn't a function that display or check data; it is the inventory data itself. Yet data is given second – class status while programming.
* In procedural programming, data type are used and worked upon by many functions. If a function makes any change to a data type, then it must be reflected to all the locations, within the program that process this data type. This is very time consuming for large sized programs.
* Procedural programming does not model real world very well.

For instance, a vehicle is an object, which is capable of moving in real world. However, the procedural programming paradigm would just be concerned about the procedure i.e. the procedure programming paradigm would just think of moving the part and not the vehicle.

OO programming: Now, the object oriented approach views a problem in terms of objects involved rather than procedure for doing it.

Object: object is an identifiable entity with some characteristics and behavior. For instance, we can say 'Orange' is an object. Its characteristics are: It is spherical shaped, its color is Orange etc. Its behavior is: it is juicy and it tastes sweet sour.

While using OOP approach the characteristics of an object are represented by its associated functions. Therefore, in Object Oriented Programming object represents an entity that can store data and has its interface through function.

How OOP overcomes procedural paradigm's problems: This RDB shortcoming is being addressed by extended relational systems[8] and middleware such as object oriented relational database gateways products Persistence[9] Now, let us see how the Shortcomings of procedural paradigm are overcome by OOP.

The object-oriented approach overcomes these shortcomings in the following manners.

* OOP approach gives data the prime consideration and by providing interface through the functions associated with it.
* An object is a complete entity i.e. it has all the data and associated functions within it. Whenever, something is to be changed for an object, only its class gets changed because it is complete in itself. All the functions that are working on this data or using it are defined within the class, they get to see the change immediately and nowhere else the change is required.

An overview of object technology: It is a basic tenet of the Object approach that "everything is an object". Some objects are immutable; examples might be integer (3,65) and character string ("Delhi", "Pune"). Other objects are mutable; examples might be the department and employee.

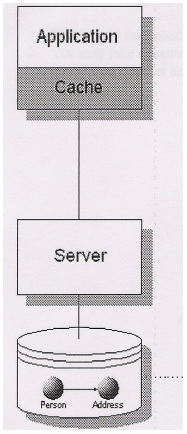
Objects are encapsulated, which means that the physical representation i.e. the internal structure of such an object, say a Dept ("department") , is not visible to users of that object; instead, user knows only that the object is capable of executing certain operations (Methods).

Creation of object oriented database: Suppose we wish to define two object classes namely DEPT (departments) and EMP (employees). Also suppose that the user-defines classes MONEY and JOB and the class CHAR is built-in. Then the necessary class definition for DEPT and EMP might look somewhat as follows:

```
CLASS DEPT
   PUBLIC (Dep#    Char,
           Dname    Char,
        Budget          Money,
           MGR       REF(EMP),
           EMPS     REF(SET(REF(REF(EMP))))----
   METHODS (HIR_EMP(REF(EMP))---code----,
           FIRST_EMP(REF(EMP))—code----,----'
CLASS EMP
    PUBLIC (EMP#                CHAR
                    ENAME       CHAR
             SALARY       MONEY
               POSITION     REF (JOB))---
METHOD (----)---;
```

Transparent persistence: Transparent persistence in object database product refers to ability to directly manipulate data stored in a database using an object oriented programming language. This is in contrast to a database sub-language used by embedded SQL or a call interface used by ODBC or JDBC. Using an object oriented database product means that you have higher performance and less code to write.

With transparent persistence, the manipulation and traversal of persistence objects are performed directly by the object oriented programming language in the same manner as in-memory. This is achieved through the use of intelligent caching as in given Fig. 1.



A person object references an address object in the object database

Fig. 1: Intelligent caching

Complex data: Complex data is often characterized by:

* A lack of unique, natural identification.
* A large number of many –to-many relation ships.
* Access using traversals.
* Frequently use of type codes such as those found in the relational schema

The discussion of complex data will use the following fragment of a clothing database that represents an XML data structure stored as objects.

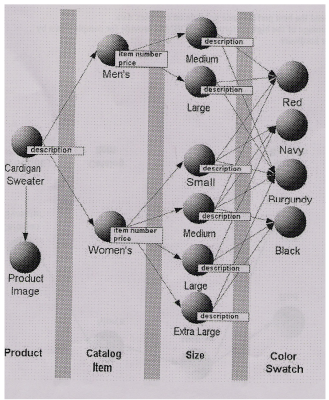


Fig. 2: Clothing database

High performance: With complex data, it is not unusual to find that an ODBMS 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.

Why are ODBMSs faster? ODBMSs are optimized for the traversals related to complex data. They also do not have any " impedance mismatch" when it comes to

using object oriented programming languages such as Java and C++. High performance can impact business considerations in two ways:

We simply may need the best performance possible on complex data. We may take advantage of the high performance ODBMSs provide for complex data by purchasing cheaper hardware.

Lack of impedance mismatch: ODBMSs allow us to store objects directly without any mapping to different data structures. RDBMSs require mapping from object to tables. This mapping to different data structures is called " impedance mismatch". The Fig. 3 shows direct storage at the left and impedance mismatch at the right.

This lack of impedance mismatch in ODBMSs give them a performance advantage over RDBMSs, especially on complex data. Impedance mismatch slow down performance on complex data because of processing needed map from one data structure (tables) to another (object).

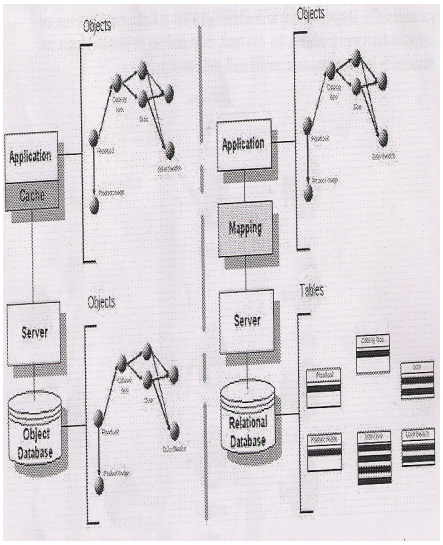


Fig. 3: Map from one data structure (tables) to another (object)

Everyday uses of object databases: We can use object database in the following:

* Pager
* Voicemail
* Flight booking
* PCs phone

Object databases are used more often than we might realize. Many times, using an object database is seen as competitive advantage and companies do not want to publicize this. As a result, object databases are invisible to users and not mentioned by companies and hence do not receive much media attention.

CONCLUSION AND FUTURE WORK

In this study we have focused on migration from RDBMS to ODBMS. We have also discussed that ODBMS is better and faster than RDBMS for complex Data. For ODBMSs, the virtue is direct manipulations of persistent objects by application software. The inseparable vices are the semantic and operational burdens attending such direct manipulation. Perhaps it is too much to ask for an application framework to support deft and natural manipulation of objects in both *off line* (RDB) and *on line* (ODBMS) forms. In any case, we offer the humble opinion that data representation issues --- the subject of much research in the academic database community --- are not the difficult problems. Instead, the core issues lie in areas long recognized to be among the most vexing of persistent data: object identity (copying vs. replication), transaction semantics (nature and lifetime of data ownership) and object naming (significance of OIDs and reference binding). Despite the cautionary tone of this paper, we are pleased with the relative success of this experiment and are encouraged to pursue several promising directions for future work. Consequently a full-fledged port and performance comparison is underway. The question thus arises: if the ODBMS port is a complete success and the RDB is retired, how will data volition be accommodated? We speculate that this dual database approach constitutes a "best of both worlds" solution. The ODBMS provides direct, fast, application-pertinent object access and the RDB provides a generalized evolution tolerant representation. The long-term solution thus may be a hybrid system, in which the ODBMS manages the live data, which is flushed to the RDB when data evolution is required.

REFERENCES

1. Codd, E.F., 1970. A Relational Model for Large Shared Data Bank, CACM.
2. Date, C.J., 1985. An Introduction to Database System, Addison Wesley.
3. Lamb, C., G. Landis, J. Orenstein and D. Weinreb, 1991. The objectstore database system. Commun. ACM, 34: 50-63.
4. Mary, E.S.L., 1995. Object Databases: The Essentials, Reading, Mass. Addison-Wesley.
5. Mary, E.S.L., 1992. ODBMS vs. Relational. J. Object-Oriented Programming Focus On ODBMS, pp: 35.
6. Brodie, M., J. Mylopoulos and J. Schmidt, 1985. On Conceptual Modeling. Springer-Verlag.
7. Atkinson, M., F. Bancilhon, D. DeWitt, K.Dittrich, D. Maier and S. Zdonik, 1989. The Object-Oriented Database System Manifesto. Proc. First Intl. Conf. Deductive and Object-Oriented Databases, Kyoto, Japan, pp: 223-40.
8. Michael, S. and G. Kemnitz, 1991. The postgres next generation database management system. Commun. ACM, 34: 78-92.
9. Arthur, M.K., R. Jensen and S. Agarwal, 1993. Persistence software: Bridging object-oriented programming and relational databases. Proc. ACM SIGMOD Intl. Conf. Management of Data, Washington DC, pp: 523-528.