OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEMS-
A NEW TOOL FOR INFORMATION CENTRES

J M JOSE
T VISWANATHAN
INSDOC
14 Satsang Vihar Marg
New Delhi 110067

To improve service efficiency and stay competitive, information centres have to adopt continuously new techniques in system development, software and hardware which move the state of the art forward in information management. Object-oriented database management systems (OODBMS) are a recent development in the field of database systems. Their adoption and use will increase the capabilities and efficiency of an information centre. OODBMSs preserve the main features of classical database management systems and at the same time eliminate their disadvantages. Inadequacies of the present data models for new applications are highlighted, the features of OODBMSs are presented and the usefulness of an OODBMS is illustrated by modelling a typical application in an information centre.

INTRODUCTION

At present, the society may be said to be in an electronic era as there is a widespread usage of electronic products such as computers, TVs, videos, etc. even by common man. Information plays a vital role in day-to-day life. The information needs of the society have undergone changes in both qualitative and quantitative terms. As a result, serving the present generation of patrons calls for information centres to adopt new techniques in system development, software and hardware. Developments in hardware and software are taking place at a rapid pace. In order to derive maximum benefits from such developments, information systems must have a flexible design. The developments in hardware have led to the availability of powerful low cost systems. Software development has been witnessing the emergence of different software paradigms such as procedural, functional, logic and object-oriented. Different paradigms are suitable for different types of applications. Exploiting this, a number of efficient third party software packages have become available in the market supporting a variety of applications.

Database technology is vital to the functions of information centres. Present database applications in an information centre vary much from that of the past. For example, there is an increasing requirement to develop application systems that can manipulate multimedia data types such as images, voice and text, manage online database systems, handling high volumes of data and provide user friendly interfaces to casual end-users. Such applications require facilities for handling complex objects, incorporating semantics into databases, and supporting powerful and efficient trigger constraint and event mechanisms. Simultaneous with the rapid changes in hardware and software technologies, expert manpower to develop new applications is becoming scarce. Hence, the software environment must permit easy development of applications by novice professionals. Conventional database management systems are not able to meet these new requirements.

WHY CONVENTIONAL SYSTEMS ARE NOT ADEQUATE?

Relational database management systems, which are most popularly used these days, have been developed to overcome the deficiencies of the earlier network and hierarchical systems. Even the relational systems fail to meet the requirements of the advanced applications of an information centre. The relational model of data presents a tabular view of the world. It assumes that every entity has a fixed number of descriptive attributes and the values of these attributes may be presented in a tabular form. Unfortunately most of the real-life problems do not fit into the tabular form very well. For example, a document database containing actual documents can not be characterised by only a set of attributes giving the description and location of the documents. A document may be composed of pictures, graphs, text, etc. To handle such a database of documents, the system should support complex extensible objects. There are other applications which demand the support of dynamic objects. Here objects may change with time and their attributes may take different values at different
FIG. 1(a) Journal

<table>
<thead>
<tr>
<th>Journal-Id</th>
<th>Title</th>
<th>ISSN</th>
<th>Vol</th>
<th>Issue</th>
<th>Year</th>
</tr>
</thead>
</table>

(Key-word: Journal-Id)

FIG. 1(b) Relational representation

<table>
<thead>
<tr>
<th>Journal-Id</th>
<th>Article-No</th>
<th>Article-Title</th>
<th>Page</th>
<th>Author</th>
</tr>
</thead>
</table>

(Key-word: Journal-Id, Article-No)
points of time. Users may inquire about states of objects corresponding to different times. This may require some sort of version control. For example, in the case of serials in a library, titles of some of them may change with time. In conventional systems, it is difficult to keep track of such history of objects.

In relational systems, non-redundant representation of objects and dependencies call for normalisation which leads to multiple tables containing attributes of the objects represented. The relationships among these multiple tables are not visible in the database structure but are stored in application programs. There is no way for a user to find out the relationships from the database and hence she needs the help of the database designer or administrator for this purpose. Due to this drawback of relational systems, the semantics or meaning of real world situations cannot be incorporated into the database. Much of the knowledge of the database designer is not captured and preserved as part of the database. As a result, those who access the database later are required to possess additional knowledge to execute for example, meaningful adhoc queries. Moreover, partitioning a real-world object into different tables leads to inconsistency problems during the process of updation and deletion. When changes are incorporated in a table, the corresponding changes need to be reflected in the related tables as well. This may not be achieved correctly always in the absence of description of the relationships in the database. If the data model is able to support complex objects directly, these problems can be avoided to a great extent.

In the practical world of database applications the difference between the end-users view of objects and that of database structure of relations is a source of great frustration and difficulty. End users find relational and other conventional representation of objects quite cumbersome, overly complicated, and difficult to understand. For example, to represent a journal containing different articles (Fig. 1(a)) in a relational system, one may have to decompose the attributes into different tables as shown in Fig. 1(b). End users want to deal with objects in a fashion that seems natural and convenient to them.

Relational systems do not support abstraction techniques such as generalisation and specialisation. For example, an entity of the type 'publication' which is a generalisation of items like books, journals, reports etc. may generally have an author and a title. But these two attributes alone do not fully characterise the entities. The general entity 'publication' needs to be specialised by different parameters for identifying the type of publication as shown in Fig. 2. For example, books do not have volume and issue numbers whereas journals do have. One has ISBN number whereas the other has ISSN number.

Object-oriented database management systems (OODBMS) is a post relational database management system which is capable of preserving the main advantages of classical database systems and eliminating their disadvantages.

OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEMS (OODBMSs)

An object-oriented database management system can be defined as a database management system with an object-oriented data model. Object-oriented data model can be perceived, in a general sense, as a framework in which the real world semantics can be expressed with ease. Instead of set oriented records as in relational systems, OODBMS supports objects. The major advantage of an OODBMS is that any real-world entity can be represented as an object. For example, in the case of a serials system in a library, the entities concerned are journals, invoices, bank drafts, reports etc. All the entities are treated as objects. Objects may be as simple as numbers or strings or as complex as the complete details about a journal. In addition to the attributes of an entity the behavior characteristics are also clubbed to the object through procedures or methods (Fig. 3). Hence, in the case of an object-oriented system, objects contain both data and procedures. Object-orientation provides a natural way of representation of real world problems. Objects having the same characteristics and behavior are grouped into a class. For example, a particular book can be considered as an object belonging to a class say BOOK. In other words, an object is an instance of a class.

Object-oriented database management systems are the result of integration of object-oriented language features with database concepts (Fig.4). Language features include object-identity, inheritance, and encapsulation. Integrating these features with database concepts such as persistence, data sharing, integrity, query etc. results in OODBMS. Object identity is a feature by which one can distinguish objects from one another, regardless of their content, location or addressability. In relational systems, records are identified using attribute values. But in
FIG: 2

```plaintext
Attributes + Method => Object
```

FIG: 3

Object Oriented Language Features + Database Features => OODBMS

```plaintext
Object Identity Inheritance Encapsulation + Persistence Sharing Integrity Query => OODBMS
```
object-oriented systems each object is identified by a unique name, which can be either system defined or user defined. This is analogous to that of distinguishing objects naturally. Encapsulation is the formal term that describes the bundling of data and procedures or methods together. As an example, in the case of a desktop publishing system, documents can be considered as the objects. They have attributes like name, type and number. The procedures associated with the documents may be 'format', 'change font', etc. When we consider a document as an object, these attributes and corresponding procedures are clubbed together in it (Fig.5). This is called the encapsulation of data and methods. These methods act as visible interfaces to the object. So, an object is like an engineering black box which produces some outputs when some inputs are given. The internal processing is hidden from the outside world. In other words, the information concerning an object are hidden from the external world, but it is identified by the functions associated with that object. It assures that an object is independent of its implementation, and this ensures data abstraction and modularity.

As in the real world, objects in a system are independent of one another and they communicate among themselves by message passing. The paradigm of communication ensures that the objects are not directly manipulated by the users. Instead, a user sends messages to an object and the object itself determines the method by which it will react.

Inheritance is the mechanism for automatically sharing data among classes and subclasses. This mechanism is very powerful and is not found in conventional systems. Inheritance allows users to create new classes by programming only the differences from the parent class. Considering the desktop publishing system, a document can be textual or graphical. The characteristics of a textual and graphical documents are different. By inheritance, we can create new classes such as, text and graphics, from a parent class document (Fig. 6). The new classes contain all the characteristics and methods of the parent class and some additional characteristics and methods. Inheritance provides a means for reducing the effort required to develop and maintain a database. But its ability to allow a database to age gracefully is far more important. The key idea in inheritance is to provide a simple and powerful mechanism for defining new classes that inherit the structure and operations from an existing class. New methods and old structure may be overridden. Polymorphism is another feature by which different objects can respond differently to the same message.

These features facilitate a sophisticated way of dealing with data and hence provide advanced methods of managing information.

MODELLING A CIRCULATION SYSTEM

Consider a real world example of a book circulation system in a library. Two classes of objects exist: the BOOK and the BORROWER. Borrower gets a book issued from a library and returns it after perusal. Borrower is a person who has a registration in the library. So BORROWER can also be considered as a class which belongs to a more general class PERSON. Hence, we can consider BOOK and PERSON as two classes of representation. A particular book is an object which belongs to the class BOOK and a particular borrower is an object which belongs to the class PERSON. The appropriate data attributes for these classes may be

PERSON = (first name, middle name, last name, address, gender, ...)
BOOK = (title, author, publisher, year, ...)

Also some of the functions associated with these classes may be get name, change name, get address, change address, etc. The attributes and associated functions together describe the class PERSON. So the class PERSON can be represented as

PERSON:
DATA:
    first name,
    middle name,
    last name,
    address,
    gender, etc.

METHODS:
    get name,
    change name,
    get address,
    change address, etc.

Thus the class PERSON has its own attributes and functions. A particular person or borrower is an instance of the class PERSON and is treated as an object having all the attributes and functions described in the class PERSON. The particular attributes of these objects will be accessible only through the functions specified with that object. For example, to change the name of a
particular person one should use the function ‘change name’ associated with the class PERSON. These functions act as visible interface to these objects and the attributes and are hidden from the outside world. Similarly, we can consider the class BOOK with its attributes and functions.

For a university library system, we can consider two categories of borrowers, one is faculty borrower and other is a student borrower. Therefore, we can consider two subclasses of the class PERSON, namely, FACULTY_BORROWER and STUDENT_BORROWER (Fig. 7). By this hierarchical representation we have captured the idea that a FACULTY_BORROWER and STUDENT_BORROWER have all the attributes such as name, address, etc. and functions, such as get name, get address, etc. specified in the class PERSON. These attributes and functions are inherited from the class PERSON. The subclasses FACULTY_BORROWER and STUDENT_BORROWER might contain some more attributes and functions, which are relevant to each of these subclasses. For example STUDENT_BORROWER might include attributes such as student_id, programme of study, batch no, etc. By this process, we have two specialized classes of class PERSON, viz. FACULTY_BORROWER and STUDENT_BORROWER. As far as class PERSON is concerned it is a generalisation of the classes STUDENT_BORROWER and FACULTY_BORROWER. There can also be subclasses of STUDENT_BORROWER such as, GRADUATE_STUDENT and RESEARCH_STUDENT, etc. The advantage of object-oriented approach is that, one can plug in any number of classes by inheriting characteristics of the existing classes at any point of the system development. This way of system design is tantamount to that of natural way of analysing real-life problems. By capturing naturally occurring objects of application domain into the classes along with the associated behavioural pattern, the object-oriented approach simplifies developmental process considerably.

CURRENT SCENARIO

A number of groups are working towards the development of OODBMS all over the world. There are about 30 OODBMS under development with commercial vendors, industrial research laboratories and universities. Most of these systems support the core object-oriented concepts and additional features with varying degrees of functionality. Some of the commercially available systems are GemStone from Servio-Logic Development Corporation, Ontos from Ontologic, Inc. and Statica from Symbolics, Inc. Some of the industrial research prototypes are IRIS from Hewlett-Packard, O2 from Altair, France, ZEITGEIST project at Texas Instruments, Jasmin project at Fujitsu, Japan, ORION from MCC Austin, Texas etc. Some of the university research prototypes are ENCORE/Observer project at Brown University, POSTGRESS at University of California and OZ+ at University of Toronto, AVANCE at University of Stockholm, Sweden etc.

CONCLUSION

Information centres require very advanced tools to meet the present day requirements. They also need a system which is easy to use even by novice users. Object-oriented database management systems are very promising for database applications of an information centre. Since OODBMS permits incorporation of more semantics into its database, it is easy to develop applications. OODBMS are likely to replace conventional systems in the coming decade.

REFERENCES