Relational Vs. Object-Oriented Development Process for Multimedia Videotex Applications

Seow-Meng Tan
University of Ottawa

Jimming Lin
University of Ottawa

Follow this and additional works at: http://aisel.aisnet.org/pacis1993

Recommended Citation
http://aisel.aisnet.org/pacis1993/14

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 1993 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Relational vs. Object-Oriented Database: 
A MIS Perspective

Seow Meng Tan
Jim Lin

University of Ottawa

Introduction

Before database management systems were developed, data resided in application-specific files and were maintained by individual applications. Consistency was the main problem as data was used and managed through different applications in different departmental data files. Database Management Systems (DBMS) were created to simplify the job of developing these applications and to ensure data consistency, availability, integrity, and ease of retrieval. Because early demands for DBMSs came from transaction-oriented users, such as accounting, customer billing services, etc., early DBMSs were designed to support alphanumerical data formatted into record-oriented files closely mapping these activities. The limitations of the early approach to database management, became obvious with the needs of users involved in such fields as Computer-Aided Design (CAD). The needs of these groups are different and they require a different kind of DBMS support than groups doing traditional data processing. This gave rise to a different kind of DBMS — Object-Oriented Database Management Systems (OODBMS).

Now, both Object-oriented and Relational Database Management Systems (RDBMS) are competing for the attention of users, with fervent supporters arguing the merits of each system. Supporters of OODBMS say that treating data types as objects can solve real-world problems more effectively than the relational model which is considered as constricting. OODBMS offers flexibility because of its ability to support abstract data types and stores code to implement services in the database itself. On the other hand, proponents of the relational model argue that its rules are necessary to keep order in a system and to maintain the purity of decision-support information.

In what follows, OODBMS will be examined and compared to today's most popular kind of traditional DBMS. In addition, issues of interest to the MIS manager in planning a corporate object-oriented database are examined in some detail.

What is Relational Database?

In the 1960s and 1970s, large databases were stored on mainframes, and the database applications were dominated by the hierarchical and network models. By the mid 1970s, the relational model, pioneered by E. Codd, gradually began to gain acceptance.[1] This kind of relational database model was originally used extensively on microcomputers, but more and more, it is now being used on larger computers for large-scale applications. Almost all of the database systems developed over the past decade were relational. Furthermore, almost all database research from the mid 1970s to mid 1980s was also based on relational ideas.[9] In fact, there can be little doubt that the relational approach represents the dominant trend in the marketplace today, and that the relational model is considered as the single most important development in the entire history of the database field.

By the early 1980s, the first systems based on the relational model appeared on the market, bringing with them the definite advantages over their predecessors. Today, a large number of relational products is available on mainframe computers, minicomputers, microcomputers, and dedicated computers (database machines), and their market is rapidly expanding and is gradually replacing the older form of database systems.

The overall objective of a relational database is to allow complex relationships between records to be expressed in a simple form.[19] When the data is presented in a simple form, users are able to use the database without being aware of the structure of the database. The success of the relational model among researchers, designers, and users is due primarily to the simplicity and the power of its concepts:[29]

- to permit a high degree of data independence;
- to furnish considerable grounds for dealing with consistency, data semantics, and redundancy problems;
- to enable the expansion of set-oriented data manipulation languages; and
- to become an extensible model which can depict and manipulate simplistic and multifarious data.

These four concepts have been achieved by the relational model and not by the hierarchical and network models due mainly to the simplicity of the relational views introducing the data in two-dimensional tables and the application of the normalization theory to database design. Relational model also utilizes relational algebra, which manipulates tables formally and simply the same mode as arithmetical operators (such as cardinal product, select, project, join, difference, union, intersection and difference) manipulate integers, uses non-procedural languages based on logical queries to specify the data one desires to acquire without having to expose how to attain them. The fourth concept, however, exists as the gist of current developments concerning extended relational models and of some work on deductive databases and on object-orientation in relational databases. The promotion of the relational model has also been aided by database language standardization, which codes the Standard Structured Query Language (SQL).[31] SQL furnishes a uniform interface to all sorts of users (end users, programmers, database administrators) for data definition, control, and manipulation.
As mentioned above, the relational model is mainly delineated by three characteristics. Thus, the data structure is simple. It uses the concept of a relation to represent a file. The relation presents data to users in a two-dimensional table, and the data is perceived by the user as tables. The operators at the user’s disposal (e.g., for data retrieval) are operators that generate new tables from old. For instance, there will be one operator to extract a subset of the rows of a given table, and another to extract a subset of the column. The relations are governed by normal forms. Normal forms are sets of increasingly stringent rules that govern the design of relations. In a relation, each column contains values of the same attribute, and each row is discrete and represents a single record. The columns hold information pertaining to the row in which it is located. Each column, also called a field, has a range of values in which it can take on, this is called its domain. An example of a record is shown below.

<table>
<thead>
<tr>
<th>Row X</th>
<th>Employee Name</th>
<th>Employee Number</th>
<th>Employee Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
<td></td>
</tr>
</tbody>
</table>

The column entitled Employee Name contains a list of names. Each column has a distinct attribute name, (for example Employee Name, Employee Number, and Employee Position.) Each row is distinct, in that one row cannot duplicate another row. All of the records are then grouped together and stored in a database file. To comply with these properties, the data must be normalized. (Normalization is the process of converting a relational database design into one that has good operational properties.)

Among the eight operators of the relational algebra are three basic operations which can be used to manipulate one or two relations in database files: select, project, and join. Select is an operation which creates a subset of records which meets a specific criteria. Join is used to relate records from separate files by matching field values. Project is used to select a subset of columns from the table. In order to control databases using these operations, query languages are used. These languages consist of a variety of commands which allow the users/programmers to indicate specific actions to perform. During the 1960s, SQL became a favourable database language for large and powerful DBMS’s such as DBase IV, Powerhouse, and Oracle. Such a structured language allows data to be distributed across heterogeneous frontends. This also allows for a more efficient and effective method for managing information. However, because technology in this area is changing so rapidly, a uniform standard has yet to be established.

There are many major strengths and major weaknesses of the relational database. The strengths can be summarized as:[4,7,8,9,11,15,19 & 22]

- ease of data retrieval;
- the ability to combine different types of information;
- simplicity of design;
- non-procedural access; and
- ease of editing.

The weaknesses of the relational database are:
- the storage capacity needed;
- the speed at which data can be retrieved;
- the data needs to be normalized and thus reduces the flexibility of the database; and
- complex objects such as a voice file cannot be stored in a relational database.

What is Object-Oriented Database?

Today, many people start to recognize the potential shortcomings of current relational DBMS and conventional programming languages. Even though traditional relational DBMS has a firm theoretical base and is capable of satisfying many applications, it still lacks important features needed for advanced applications, such as abstract data types, complex integrity constraints, and versioning.[34] Now, object-oriented data model can produce a new kind of database that matches real-world applications more closely, and avoids the normalization problems often associated with relational DBMS.

Unfortunately, the term object-oriented database management system (OODBMS) is still not very well defined as it starts to emerge in the commercial and research world. It often means different things to different people as a result of what one emphasizes: the database or object-oriented programming language. This new technology was invented in the mid 1980s and is still considered by many as immature, and still suffers from the lack of standards and unresolved design issues that relational DBMS did almost a decade ago.

Nevertheless, object-oriented database (OODB) represents a new philosophy and implementation for storing information and data to users. It constitutes the technology that combines the powerful characteristics of object-oriented systems with traditional database capabilities by integrating an object-oriented programming language (such as C++ or Smalltalk) to build a high performance, well structured, easily maintained data application.[3] The main idea behind OODB is to deal operational data as objects that more closely resemble their counterparts in the real world instead of dealing with computer-oriented constructs such as records and fields.[5] Objects have various natural interrelationships reflecting real-world ones and paralleling relationships in a relational database.

The advantages of this direct representation capability in data modelling and programming are obvious. Object-oriented concepts and tools enable technology, permitting real-world problems, to be declared readily and naturally. It is designed to handle any type of complex information and real world data, such as graphics, voice, information, rows and columns, file elements, bit map, and images.[48] Object-oriented techniques also furnish the rudimentary methodologies to erect complex software systems out of modularized reusable software units, and is ideal for deeply complicated data structures that are not handled and achieved easily by relational databases.[46]

An object is a thing that exists and has identity. Examples for objects are items such as the chair in the corner, a house, or a man. A group of similar objects form an object class. For instance, chair, room and people are examples of object classes. An object is an instance of an object class described by attributes. Objects in OODB are defined in hierarchies of classes and sub-classes which inherit attributes and behaviours of parent classes; instances of classes contain specific attribute values and carry out certain functions. These functions or behaviours are carried out in responses to requests from other objects; responses to message change object attribute values and/or return the data to the user. Objects also allow a message to refer to objects from more than one class, and dynamic binding, which produces a response unique to the type of object being referred to a message.[20]

As mentioned above, OODB combines an object-oriented programming language with the functionality of a database system. However, in order to understand its characteristics, it is necessary to examine the fundamental concepts of object-oriented programming systems (OOOPS).

In defining an object-oriented database, the object-oriented modelling technique (OMT) is used.[2] This technique displays three distinct characteristics:
Encapsulation (Abstract Data Type): Data and functions that share similar characteristics are placed within the same class. A object-oriented database encapsulates data with procedures which act upon the data, into packages called objects. An object is a thing that exists and has identity (e.g., a chair, a room). Similar objects can be grouped together to form a class. These separately identifiable objects (objects), are linked by three types of relationship: generalization, aggregation, and association. A generalization relationship simply partitions a class into mutually exclusive subclasses. Aggregation relationships combine low-level objects into composite aggregates. An association relates two or more independent objects. Examples of these relationships are shown below.

Inheritance: objects can inherit the characteristics of other objects within its class. This means that data may be "inherited" or retrieved from previously stored fields. Objects may be data objects or view objects (which are merely representations of data objects). An object is a thing that exists and has an identity. For example, a chair in the corner of a room or room #102. An object class is a group of similar objects such as chair, room, people. Classes of objects or inheritance is achieved through libraries. All program entities are instances of classes. All objects are stored in class structures determined by methods (or functions), which yields the benefits of inheritance and reusability. Through inheritance, new software modules (i.e., classes) can be created on top of an existing hierarchy of modules. Code sharing (and thus reusability) among software modules is enabled by inheriting behavior. The combination of these two types of inheritance renders a powerful modelling and software development strategy. The essence of inheritance hierarchies is that object types inherit most of their attributes from generic or less specialized types.

Interaction (Object Identity): the result of sending a message between objects involving a function depends on the class of the object at run-time. The object space is constructed on top of basic or rock-bottom objects (characters, numbers, booleans, integers, floating point, etc.).

Through supporting object identity and inheritance, OODB allows structuring and reference sharing of objects. Referential sharing is where multiple applications, products, or objects share a common sub-object. The other type of sharing is common concurrent sharing of objects whereby the majority of the functionality of database management systems (especially on main’s and mainframe) is devoted to controlling concurrent circles to persistent database objects by multiple users or applications.

In addition to the elements mentioned above (abstract data types, inheritance and object identity), OODB also support the following interrelated database concepts and capabilities: persistence, transactions, concurrency control, recovery, querying, versioning, integrity, security and performance.[25]

To delineate a set of objects with the same representation, data types are utilized. Associated with each data type is a number of operations. Abstract data type (ADT) extends the notion of a data type through hiding the implementation of the user-defined operations (messages) associated with the data type. Languages supporting ADT furnish constructs to directly define data structures and the operations employed to manipulate occurrences (instances) of the data structures. Furthermore, all manipulations of instances of the data type are realized through operations linked with the data type. A language that supports abstract data typing will permit the instances of data type to be manipulated only through a prescribed aggregation of operations associated with the type.

All these procedures are central. In object-oriented systems, a collection of independent objects communicate with each other through procedures is often called a message. The objects are the active entities and the procedures are the passive entities that are passed from one object to another. Thus, the objects, not the procedures, are central. Commonly used procedures are: forms, reports and queries. These procedures are saved in libraries or data dictionaries. Each procedure becomes a self-contained, modular object that contains information about itself. These objects can respond to other objects or to the user via messages it receives. Objects can be modified copied and reused. They are always ready and waiting for input, hence object-oriented programming systems are sometimes called event-driven systems.

For programmers, newly created objects are stored in libraries and become part of the language. Objects can then be modified, copied, and reused. As well, any changes made to data or its related operations, is done within the object itself, the application need not be changed. This makes up-dating and systems maintenance a lot less painful.[25]

Object-oriented user interface, otherwise known as Graphic User Interface, also provides users easier interaction with complex resources.[37] Users can use icons to manipulate computer resources and data. Several products on the market such as Hewlett-Packard's Neuvor, Steve Jobs' Next computer, Microsoft Windows and the Macintosh desktop environment are of this kind.

The major advantages of object-oriented database can be summarized as[1,5,6,28,32,33,37]:
- the ability to handle complex data at high speeds (up to 1000 times faster at graphics and documentation than relational systems);
- the ability to store unstructured and complex information;
- the ability to create new objects, while keeping the changes to the objects;
- the ability to process other objects uniformly — polymorphism;
- the ability to access data items without the use of primary keys and fields;
- the ability to adapt to individual needs by integrating individual pieces into complete design;
Abstract pointers are used in OODB to help implement relationships among objects whereas relational systems use keys to join operations. A relational system cannot get hold of an entire complex object in one query. While an object-oriented database cannot be queried, the OODB can retrieve an object composed of many other objects as a whole. That is why rules must be added separately to RDBS or to other applications that built on them.

Rules are part of the structure of OODB as well. OODB is not as mathematically well-defined as relational databases but they are a good choice for data structures not easily handled by the relational model. OODB supports abstract data types, whereas RDBS is not capable of storing objects. OODB appears to support inherently more complex than the select, project and join commands used in relational DBMS.

On the other hand, supporters of RDB argue that OODB lacks the rules necessary to ensure information integrity because OODB lacks the mathematical elegance of RDBS. As well, OODB is becoming too difficult to control when the project is large. However, current trends indicate a more unified approach. Most database vendors are beginning to add object-oriented features to their databases in order to improve the user interface, and to facilitate the storage and retrieval for graphics, text and voice. These new databases have been called Third Generation Databases [46,48] and consist mainly of extensions to SQL to allow for object data types. For example, Omos supports objects and SQL, and Interface supports the use of objects in its relational model by using Binary Large Objects (BLOB).

MIS Concerns When Planning a Object-Oriented Database

To be successful with object-oriented technology in planning corporate databases, MIS managers must first think carefully about what they want from object-oriented programming (OOP) and examine critically whether it is capable of delivering the desirable output. It will be a mistake to think that any organization which adopts new technology (in this case OODBMS) will automatically reap wonderful benefits and move the organization into high gear. It is a dangerous notion that if one is doing OOP, a product will emerge without any effort.[10]

Ideally, the desirable OODB must be able to support shared object identity, and must be able to respond dynamically to changes to the data and its structure. It is often very wise to start small, so that if mistakes are made, they will not have bled the company.

When planning for a new OODBMS, MIS managers must be prepared to manage their projects differently than they used to in the past. During a project, every software team should follow a common style. Therefore, it is the responsibility of the MIS managers to develop new rules, guidelines and standards of how things should be done. For example, a few simple rules in terms of how variables and objects are named should be developed, so that programmers will be able to read each other's code more efficiently and will not misunderstand the relationships between object types.

It is also a good idea to assign a small project group to think about the libraries and classes being developed in the application. It is important that the knowledge that goes into this design work must be captured on paper. As well, someone else should be designated to ensure the software makes sense for the next project as to enforce extendability and continuity. If an object-oriented database is desired, following such steps will make a difference towards the bottom line. Besides, it provides intangible benefits by contributing to employee satisfaction.

MIS managers must realize that object-oriented programming is a relatively new concept to many (even though it was invented in the mid 1950s), and should be aware of its helpfulness and limitations in developing new applications. For example, a company wishes to develop an Expert System, or Decision Support Systems, then object-oriented may be a potential direction to take.
An overview of the differences between the two models is summarized in Table 1.

<table>
<thead>
<tr>
<th>RELATIONAL DATABASES</th>
<th>OBJECT-ORIENTED DATABASES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage of Data</td>
<td></td>
</tr>
<tr>
<td>Constricting: all data must be stored in fixed-length records</td>
<td>Flexible; can directly store and retrieve objects as a whole, rather than trying to fit everything into records</td>
</tr>
<tr>
<td>Operational Data</td>
<td></td>
</tr>
<tr>
<td>Restricts data to a table format — rows and columns</td>
<td>Objects may consist of tests, programs, scanned images, video, rows and columns, graphics, finite elements, and bit-map</td>
</tr>
<tr>
<td>Data Manipulation</td>
<td></td>
</tr>
<tr>
<td>Operations include: Select, Join, Project</td>
<td>Can support operations more complex than the Select, Join, and Project commands</td>
</tr>
<tr>
<td>Join Queries</td>
<td></td>
</tr>
<tr>
<td>Uses keys to &quot;join&quot; operations</td>
<td>Uses abstract pointers to implement relationships among objects</td>
</tr>
<tr>
<td>Execution of Instructions</td>
<td></td>
</tr>
<tr>
<td>Takes many instructions to retrieve a record out of a field</td>
<td>Requires fewer instructions</td>
</tr>
<tr>
<td>Data Definition</td>
<td></td>
</tr>
<tr>
<td>Database must be created using a table — not a natural way of thinking</td>
<td>Objects more closely resemble real world information</td>
</tr>
<tr>
<td>Database Architecture</td>
<td></td>
</tr>
<tr>
<td>Database is made up of rows (records) and columns (fields)</td>
<td>Database is made up of capsules of data and codes — objects</td>
</tr>
<tr>
<td>Generality</td>
<td></td>
</tr>
<tr>
<td>Uses computer oriented constructs: primary keys, records, and fields</td>
<td>Does not use generally understood constructs, but rather encapsulated data</td>
</tr>
<tr>
<td>Data Reference</td>
<td></td>
</tr>
<tr>
<td>Individual objects have names</td>
<td>Individual objects do not have names, rather, it is class that have names</td>
</tr>
<tr>
<td>Data Category</td>
<td></td>
</tr>
<tr>
<td>Data type</td>
<td></td>
</tr>
<tr>
<td>Standard Query Language (SQL)</td>
<td>Classes</td>
</tr>
<tr>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>C++, Object Pascal</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of Relational and Object-Oriented

In addition, MIS manager must consider and understand the three important aspects of object-oriented programming:

- **Environments:** Traditionally, in order to plan effectively for corporate databases, managers must take into account five main areas. First, there is the size and structure of the database. From a technical point of view, the database must be designed to accommodate the required number of data elements, the size of the fields, the volume of transactions, and the response time required for on-line data entry. The nature of the transactions and the requirement for file updating will help to determine the file structure. Workstations are the second consideration. The flow of the transactions and the number of personnel involved are two of the requirements that will determine the type, number, and placement of workstations. Third, there are data security measures. These will depend on the required confidentiality of the information and the current and possible level of physical security. Password protection to restrict access to the entire database, password protection of individual files and databases, and the physical security of the equipment itself will all have to be considered in the system design. Fourth, there are problems associated with archiving and updating files. If files are updated on-line in real time, updating becomes mainly a software and data storage problem. If there is some combination of on-line and batch updating specific procedures, software, and equipment may be needed. The fifth consideration is data communication and interfaces with other systems. The design for database systems should state explicitly what equipment and procedures will be used to meet data communications requirements. This involves both local in-office equipment and the data communications equipment and capabilities of the host computer. If this in-office equipment is communicating with a host computer at some remote location. The design for interfaces to other systems should address the possibility that transaction files will be used as part of a larger database with other files containing related information. In such instances, data fields and records should be designed to facilitate integration with other files. Another type of
interface is between a transaction processing system and information analysis system. In this area, the designer should know something about the standard reports and queries that managers and other personnel would want from the transaction database. Another major concern for the MIS manager is ideas of distributed systems. A recent trend to support such systems has been through the usage of a Local Area Network (LAN). The MIS manager must realize if a LAN is an inevitable part of the corporation. If so, he/she must ensure that the terminal front-ends are compatible with the LAN server. This will allow front-ends such as Dbase IV, Powerhouse etc., to share data seamlessly.[33]

**Database Models:** An important issue to consider is the ability of current offerings to integrate with existing applications and data structures, and to accommodate new products from other vendors in the future. In particular, attention should be paid to the degree of forward and backward compatibility offered by application development tools. A global join facility that allows an application to work with data in different file structures simultaneously should be considered.[6]

**Software Development & Language Tools:** Some existing tools today such as Microsoft Windows already displays an object-oriented front-end. Here the user sees each application as a separate object, and can move from one to the other with a simple click of a mouse. Today, there are many software engineering tools for object-oriented languages (for example, like Computer Aided Software Engineering). These tools help programmers/users to define their objects to help improve system performance and development cost and time.

Moreover, it is important to point out that there are still many great issues to take into account when planning for a corporate OODB. Other issues such as:

- What the database will be intended and used for?
- Will it be used mainly for mathematical applications, or is it for scientific or engineering purposes, or is it for commercial purposes only?
- Who will be using the database? Will the end users be programmers, managers, computer operators, or perhaps all of the above?
- How involved will the users need or want to be in database design?
- What type of performance is required and desired? Does the structure of the database promote the availability of the data? (Can users quickly retrieve and up-date relevant data?)
- Is price/performance a major issue, or is throughput more important? Is cost a major development constraint?
- How closely must the database model the real world concepts?
- How important is it that the database be easily changeable, portable, compatible with existing systems? How easily can the database be extended to new applications without disrupting on going work?
- How the issues of security and data integrity should be addressed?
- How coherent is the structure of the database to end-users, other architects, and the original designers after a period of time?
- Will it be necessary to include other resources, such as expert systems as a part of or in conjunction with the database.[28]

**Is There Room For Both RDB and OODB?**

Already there is concrete evidence to some experts that the technology brought by OODBMS is not the revolutionary change (in database technology) that was once anticipated —- instead, it is evolutionary. If a user has no need for overly complex data, then a relational model can easily perform as much as would be required by a object-oriented model. However, some would disagree as they defend object-orientation.[22]

Because relational approach is already so widely accepted, it would be difficult to motivate users to change to object-orientation. Therefore, it may be ideal to target the OODBMS towards a market which is not currently involved in RDBMS. This is obviously the marketing approach for Olognic Corporation.

Unless a programming language such as object-oriented COBOL enters the market, it seems that there will always be a need for both relational and object-oriented databases. It is not a question of whether there is room enough or not for both techniques but rather if one technique will survive longer than the other. At present, it seems that there is a need for both.

**The Future of Database: Is OODB the Paradigm for the 1990s?**

Even though RDB is not yet overwhelmed by OODB, it appears that OODB is providing the RDB with some healthy competition. It offers increased flexibility, ease of use, easy maintainability and changeability, all while promising quick systems and increased programmer productivity. Some believe that OODB will be the wave of the future while others disagree. Indeed, object-oriented technology is taking off in many organizations, particularly in scientific and engineering firms. OODB offers the advantage of having a standard language interface (usually C ++), while relational database still lacks this standard. (SQL standards in RDBMS still do not seem to be very strictly adhered to.) There are, however, some problems with OODB that need to be addressed before it can penetrate the business world that RDB already did. First, many consider the OODB's dependence on C ++ or Object C a major handicap. They believe that a database should be able to run different types of application (such as C language, FORTRAN, Lotus 1-2-3, etc) and thus should not be too reliant on one particular language. There is also the need for standardization. The two major object-oriented programming languages, namely C ++ and Smalltalk are based on entirely different concepts. Until this day, there is still no general consensus on what standards would be appropriate.

OODB may also have trouble penetrating the market due to resistance to change. For example, if users are certain that they need SQL, then OODB will have to provide SQL, which in turn slows down the performance. The learning curve for OODB is considered high, and people are reluctant to go from using a system that they already know and understand so well to using one that they will have to spend a great deal of effort to learn.[34] By the same token, resistance to change might also come from the fact that many companies have already sunk in ten of thousands of dollars (or perhaps even more) in developing RDB in the past decade. To many of them, throwing away software which is so expensive and workable is just not a very sensible thing to do nowadays.

Nevertheless, advocates of the RDB who believe that there is no place for the OODB in the 1990s will have a rude awakening. It is already apparent that OODB is providing real competition for the RDB. The fact that many RDB vendors are now integrating object-
oriented ideas into RDB is the proof of the need and demand for object-oriented technology. It seems that vendors and users are becoming aware of the need for alternative to the RDB. The near future of this industry is likely to see an increasing number of RDB with object-oriented extensions as OOP promises notable gains in productivity. When design standards are set and better development tools are available, OOP will dramatically simplify network computing.

Although OODB may not replace RDB as the industry standard in the near future, they will definitely provide some well needed features and functions in RDB in the long run. Because RDB is more normalized and standardized with its use of relational mathematical models, it would be ideal to incorporate these advantages into OODB. In turn, the flexibility of OODB would be beneficial to the functioning of a combined system.

Conclusion

It is widely believed that the direction of DBMS is toward intelligent connectivity, and the demand for either RDB or OODB applications (or perhaps a combination of both) will no doubt continue to grow in the 1990s. Developers and users will have access to distributed data without compromising the power of their desktop computers. Therefore, MIS manager must develop a sound knowledge of database structures and program logic, as well as the Object Modelling Technique (OMT) or other database modelling techniques. OODB will also be vital for managing objects distributed across networks (with object directories in networks). OODB promises significant gains in software functionality, programming productivity and simplification of network computing for users. However, as with any innovation, there are obstacles to overcome such as the need for standards (a consensus based on concepts and ideas) and the need for more rigorous development models and design standards. The question remains whether MIS manager will adapt a RDBMS over a OODBMS, or vice-versa. After all, whatever the preference or decision, the ultimate goal of database modelling is still to design a better database.

References


