



BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2102

DATABASE MANAGEMENT SYSTEM

Unit 13

Object Oriented Database Management System

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction			3 – 4
	1.1 Objectives			
2	Next Generation Data Base System		1	5
3	New Database Application		1	6 – 7
4	Object Oriented Database Management System		1	8 – 10
5	Features of Object Oriented System		1	11
6	Advantages of Object Oriented Database Management System		1	12 - 16
7	Deficiencies of Relational Database Management System		1	17 – 18
8	Difference between Relational Database Management System and Object Oriented Database Management System	I	1	19 – 21
9	Alternative Object Oriented Database Strategies		1	22 – 23
10	Summary			24
11	Terminal Questions			24
12	Answers			25 – 26

1. INTRODUCTION

Since the 1960s, Data Base Management Systems (DBMS) have been widely used in the data processing environment. The support of characteristics such as data sharing, independence, consistency, and integrity is the main reason for its success which traditional file management system does not inherently offer.

A database system is usually organized according to a data model. In Unit 3, we discussed the three most popular models: hierarchical, network, and relational. The difference among all these three models is in the way of organizing records, although they are record-based. They were mainly designed to process a large amount of relatively simple and fixed format data. DBMS, based on these models along with sophisticated indexing and query optimization techniques, have served business-oriented database applications especially well.

RDBMSs were originally designed for mainframe computer and business data processing applications. Moreover, relational systems were optimized for environments with a large number of users, who issue short queries. But today's application has moved from centralized computer-aided design (CAD), multimedia systems, software engineering, and knowledge database. These operations require complex operations and data structure representation. For example, a multimedia database may contain variable length text, graphics, images, audio, and video data. Finally, a knowledge base system requires data-rich in semantics.

Existing commercial DBMS, both small and large have proven inadequate for these applications. The traditional database notion of storing data in two- dimensional tables, or in flat files, breaks down quickly in the face of complex data structures and data types used in today's applications.

Research in modeling and processing complex data has gone in two directions:

- a. Extending the functionality of RDBMS
- b. Developing and implementing OODBMS that is based on object oriented programming paradigm.

OODBMSs are designed for use in today's application areas such as multimedia, CAD, office automation, etc. In this unit, we will touch upon some of the basic issues related to OODBMS.

1.1 Objectives

By the end of this unit, you should be able to:

- ❖ *Define what is object oriented DBMS*
- ❖ *Differentiate between RDBMS and OODBMS*
- ❖ *List next-generation database systems*
- ❖ *List advantages of object oriented DBMS*



2. NEXT GENERATION DATABASE SYSTEM

Computer science has gone through several generations of database management starting with indexed files and later, network and hierarchical database management systems (DBMS). More recently, relational DBMS revolutionized the industry by providing powerful data management capabilities based on a few simple concepts. Now, we are on the verge of another generation of database systems called Object oriented DBMS, based on object oriented programming paradigm. This new kind of DBMS, unlike previous DBMS models, manages more complex kinds of the database management system (KDBMS) which is used to support the management of the shared knowledge. It supports a large number of complex rules for automatic data inferencing (retrieval), and maintenance of data integrity.

The goal of this new DBMS is to support a much wider range of data-intensive applications in engineering, and graphic representation – scientific and medical. This new DBMS can also support new generations of traditional business applications.

Self-Assessment Questions - 1

1. OODBMS is an acronym for_____.
2. KDBMS is an acronym for_____.
3. The new DBMS can support new generations of traditional business _____.

3. NEW DATABASE APPLICATION

Some applications that require the manipulation of large amounts of data can benefit from using a DBMS. However, the nature of the data in these applications does not fit well into the relational framework.

1. **Design databases:** Engineering design databases are useful in computer-aided design/manufacturing/software engineering (CAD/CAM/ CASE) systems. In such systems, complex objects can be recursively partitioned into smaller objects. Furthermore, an object can have different representations at different levels of abstraction (equivalent objects). Moreover, a record of an object's evolution (object versions) should be maintained. Traditional database technology does not support the notions of complex objects, equivalent objects, or object versions.
2. **Multimedia databases:** In modern office information or other multi-media systems, data includes not only text and numbers but also images, graphics, and digital audio and video. Such multimedia data is typically stored as sequences of bytes with variable lengths, and segments of data are linked together for easy reference. The variable-length data structure cannot fit well into the relational framework, which mainly deals with fixed-format records. Furthermore, applications may require access to multimedia data on the basis of the structure of a graphical item or by following logical links. Conventional query languages were not designed for such applications.
3. **Knowledge bases:** Artificial intelligence and expert systems represent information as facts and rules that can be collectively viewed as a knowledge base. In typical Artificial Intelligence applications, knowledge representation requires data structures with rich semantics that go beyond the simple structure of the relational model. Artificial decomposition and mapping would be necessary if a relational DBMS were used. Furthermore, operations in a knowledge base are more complex than those in a traditional database. When a rule is added, the system must check for contradiction and redundancy. Such operations cannot be represented directly by relational operations, and the complexity of checking increases rapidly, as the size of the knowledge base grows.

In general, these applications require the representation of complex data elements as well as complex relationships among them. Users in these environments have found relational technology inadequate in terms of flexibility, modeling power, and efficiency.

Self-Assessment Questions - 2

4. CASE is an acronym for_____.
5. The_____data is typically stored as sequences of bytes with variable lengths, and segments of data are linked together for easy reference.
6. Artificial intelligence and expert systems represent information as facts and rules that can be collectively viewed as a_____.



4. OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM

Object-oriented technologies in use today include object-oriented programming languages (e.g., C++ and small talk), object-oriented database systems, object-oriented user interfaces (e.g., Macintosh and Microsoft Windows systems), and so on. An object-oriented technology is a technology that makes available to the user's facilities that are based on object-oriented concepts. To define object-oriented concepts, we must first understand what an object is.

Object

The term object means a combination of data and program that represents some real-world entity. For example, consider an employee named Amit; Amit is 25 years old, and his salary is \$25000. Then Amit may be represented in a computer program as an object. The data part of this object would be (name: Amit, age: 25, salary: \$25000). The program part of the object may be a collection of programs (hire, retrieve the data, change age, change salary, fire). The data part consists of data of any type. For the Amit object, the string is used for the name, integer for age, and monetary for salary; but in general, even any user-defined type, such as Employee, may be used. In the Amit object, the name, age, and salary are called attributes of the object.

Encapsulation

Often, an object is said to encapsulate data and program. This means that the users cannot see the inside of the object, but can use the object by calling the program part of the object. This is not much different from procedure calls in conventional programming; the users call a procedure by supplying values for input parameters and receive results in output parameters.

Inheritance and Class

The term object-oriented roughly means a combination of object encapsulation and inheritance. The term inheritance is sometimes called reuse. Inheritance means roughly that a new object may be created by extending an existing object. Now let us understand the term inheritance more precisely. An object has a data part and a program part. All objects that have the same attributes for the data part and the same program part are collectively called a class (or type). The classes are arranged such that some classes may inherit the attributes and program parts from some other classes.

Amit, Ankit, and Anup are each an Employee object. The data part of each of these objects consists of the attributes Name, Age, and salary. Each of these Employee objects has the same program part (hire, retrieve the data, change age, change salary, fire). **Each program in the program part is called a method.** The term class refers to the collection of all objects that have the same attributes and methods. In our example, the Amit, Ankit, and Anup objects belong to the class Employee, since they all have the same attributes and methods. This class may be used as the type of an attribute of any object. At this time, there is only one class in the system namely, the class Employee; and three objects that belong to the class namely Amit, Ankit, and Anup objects.

Inheritance Hierarchy or Class Hierarchy

Now suppose that a user wishes to create two sales employees, Jai and Prakash. But sales employees have an additional attribute namely, commission. The sales employees cannot belong to the class Employee. However, the user can create a new class, Sales Employee, such that all attributes and methods associated with the class Employee may be reused and the attribute commission may be added to Sales Employee. The user does this by declaring the class Sales Employee to be a subclass of the class Employee. The user can now proceed to create the two sales employees as objects belonging to the class Sales Employee. The users can create new classes as subclasses of existing classes. In general, a class may inherit from one or more existing classes, and the inheritance structure of classes becomes a directed acyclic graph (DAG); but for simplicity, the inheritance structure is called an inheritance hierarchy or class hierarchy.

The power of object-oriented concepts is delivered when encapsulation and inheritance work together.

Since inheritance makes it possible for different classes to share the same set of attributes and methods, the same program can be run against objects that belong to different classes. This is the basis of the object-oriented user interface that desktop publishing systems and windows management systems provide today. The same set of programs (e.g., open, close, drop, create, move, etc.) apply to different types of data (image, text file, audio, directory, etc.).

If the users define many classes, and each class has many attributes and methods, the benefit of sharing not only the attributes but also the programs can be dramatic. The attributes and

programs need not be defined and written from scratch. New classes can be created by adding attributes and methods of existing classes, thereby reducing the opportunity to introduce new errors to existing classes.

Self-Assessment Questions - 3

7. The term_____means a combination of data and program that represents some real-world entity.
8. The_____means that the users cannot see the inside of the object but can use the object by calling the program part of the object.
9. The term inheritance is sometimes called_____.
10. Each program in the program part is called a_____.
11. In general, a class may inherit from one or more existing classes and, the inheritance structure is called an inheritance hierarchy or_____.



5. FEATURES OF OBJECT ORIENTED SYSTEM

Object-oriented systems make these promises:

Reduced maintenance: The primary goal of object-oriented development is the assurance that the system will enjoy a longer life while having far smaller maintenance costs. Because most of the processes within the system are encapsulated, the behaviors may be reused and incorporated into new behaviors.

Real-world modeling: Object-oriented systems tend to model the real world in a more complete fashion than do traditional methods. Objects are organized into classes of objects, and, objects are associated with behaviors. The model is based on objects rather than on data and processing.

Improved reliability: Object-oriented systems promise to be far more reliable than traditional systems, primarily because new behaviors can be built from existing objects.

High code reusability: When a new object is created, it will automatically inherit the data attributes and characteristics of the class from which it was spawned. The new object will also inherit the data and behaviors from all superclasses in which it participates.

processing.

Self-Assessment Questions - 4

12. The primary goal of object-oriented development is the assurance that the system will enjoy a longer life while having far smaller _____.
13. The Real World Model is based on objects rather than on _____ and processing.
14. Object-oriented systems promise to be far more _____ than traditional systems.

6. ADVANTAGES OF OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM

An object-oriented programming language (OOPL) provides facilities to create classes for organizing objects, create objects, structure an inheritance hierarchy to organize classes so that subclasses may inherit attributes and methods from superclasses, and call methods to access specific objects. Similarly, an object-oriented database system (OODB) should provide facilities to create classes for organizing objects, create objects, structure an inheritance hierarchy to organize classes so that subclasses may inherit attributes and methods from superclasses, and call methods to access specific objects. Beyond these, an OODB, because it is a database system, must provide standard database facilities found in today's relational database systems (RDBs), including nonprocedural query facility for retrieving objects, automatic query optimization, and processing, dynamic schema changes (changing the class definitions and inheritance structure), automatic management of access methods (e.g., B+-tree index, extensible hashing, sorting, etc.) to improve query processing performance, automatic transaction management, concurrency control, recovery from system crashes, and security and authorization. Programming languages, including OOPLs, are designed with one user and a relatively small database in mind. Database systems are designed with many users and very large databases in mind; hence performance, security and authorization, concurrency control, and dynamic schema changes become important issues. Further, transaction systems are used to maintain critical data accurately; hence, transaction management, concurrency control, and recovery are important facilities.

In view of the fact that C++, despite its growing popularity, is not the only programming language that database application programmers are using or will ever use, there is a significant gulf between a programming language and a database system that will deliver the power of object-oriented concepts to database application programmers. Regardless of the approach, OODBs, if done right, can bring about a quantum jump in the productivity of database application programmers and even in the performance of the application programs.

Advantages of Object-Oriented Databases

Systems developed with object-oriented languages have many benefits, as previously discussed. Yet, as also described, these systems have particular attributes that can be complemented with object-oriented databases. These attributes include lack of persistence, inability to share objects among multiple users, limited version control, and lack of access to other data, for example, data in other databases.

In systems designed with object-oriented languages, objects are created during the running of a program and are destroyed when the program ends. Providing a database that can store the objects between runs of a program offers both increased flexibility and increased security. The ability to store the objects also allows the objects to be shared in a distributed environment. An object-oriented database can allow only the actively used objects to be loaded into memory, and thus minimizes or preempts the need for virtual memory paging. This is especially useful in large-scale systems. Persistent objects also allow objects to be stored for each version. This version control is useful not only for testing applications but also for many object-oriented design applications where version control is a functional requirement of the application itself. Access to other data sources can also be facilitated with object-oriented databases, especially those built as hybrid relational systems, which can access relational tables as well as other object types.

Object-oriented databases also offer many of the benefits that were formerly found only in expert systems. With an object-oriented database, the relationships between objects and the constraints in objects are maintained by the database management system, that is, the objects themselves. The rules associated with the expert system are essentially replaced by the object schema and the methods. As many expert systems currently do not have adequate database support, object-oriented databases afford the possibility of offering expert system functionality with much better performance.

Object-oriented databases offer benefits over current hierarchical and relational database models. They enable support of complex applications not supported well by the other models. They enable programmability and performance, improve navigational access, and simplify concurrency control. They lower the risks associated with referential integrity, and they provide a better user metaphor than the relational model.

Object-oriented databases by definition allow the inclusion of more of the code (i.e. the object's methods) in the database itself. This incremental knowledge about the application has a number of potential benefits for the database system itself, including the ability to optimize query processing and to control the concurrent execution of transactions.

Performance, always a significant issue in system implementation, may be significantly improved by using an object-oriented model instead of a relational model. The greatest improvement can be expected in applications with high data complexity and large numbers of inter-relationships. Clustering, or locating the related objects in close proximity, can be accomplished through class hierarchy or by other interrelations. Caching, or the retention of certain objects in memory or storage can be optimized by anticipating that the user or application may retrieve a particular instance of the class. When there is high data complexity, clustering and caching techniques in object databases gain tremendous performance benefits that relational databases, because of their fundamental architecture, will never be able to approach.

Object-oriented databases can store not only complex application components, but also larger structures. Although relational systems can support a large number of tuples (i.e., rows in a table), individual types are limited in size. Object-oriented databases with large objects do not suffer performance degradation, because the objects do not need to be broken apart and reassembled by applications, regardless of the complexity of the properties of the application objects.

Since objects contain direct references to other objects, complex data sets can be efficiently assembled using these direct references. The ability to search by direct references significantly improves navigational access. In contrast, complex data sets in relational databases must be assembled by the application program using the slow process of joining tables.

For the programmer, one of the challenges in building a database is the data manipulation language (DML) of the database. DMLs for relational databases usually differ from the programming language used to construct the rest of the application. This contrast is due to differences in the programming paradigms and mismatches of type systems. The programmer must learn two languages, two toolsets, and two paradigms because neither

alone has the functionality to build an entire application. Certain types of programming tools, such as application generators and fourth-generation languages (4GLs) have emerged to produce code for the entire application, thereby bridging the mismatch between the programming language and the DML, but most of these tools compromise the application programming process.

With object-oriented databases, much of this problem is eliminated. The DML can be extended so that the application can be written in the DML. Or an object-oriented application language, for example, C++ can be extended to be the DML. More of the application can be built into the database itself. Class libraries can also assist the programmer in speeding the creation of databases. Class libraries encourage the reuse of existing code and help minimize the cost of later modifications. Programming is easier because the data structures model the problem more closely. Having the data and procedures encapsulated in a single object makes it less likely that a change to one object will affect the integrity of other objects in the database. Concurrency control is also simplified with an object-oriented database. In a relational database, the application needs to look into each record in each table explicitly because related data is re-represented across a number of tables. Integrity, a key requirement for databases, can be better supported with an object-oriented database because the application can lock all the relevant data in one operation. Referential integrity is better supported in an object-oriented database because the pointers are maintained and updated by the database itself. Finally, object-oriented databases offer a better user metaphor than relational databases. The tuple or table, although enabling a well-defined implementation strategy, is not an intuitive modeling framework, especially outside the domain of numbers. Objects offer a more natural and encompassing modeling metaphor.

Self-Assessment Questions - 5

15. Programming languages, including OOPLs, are designed with one user and a relatively small _____ in mind.
16. Providing a database that can store the objects between runs of a program offers both increased flexibility and _____.
17. Object-oriented databases also offer many of the benefits that were formerly found only in _____.
18. Object-oriented databases offer benefits over current _____ and _____ database models.
19. Object-oriented databases can store not only complex application components but also _____.



7. DEFICIENCIES OF RELATIONAL DATA BASE MANAGEMENT SYSTEM

The data type facilities are the keys to eliminating three of the important deficiencies of RDBs. These are summarized in the paragraphs given below.

RDBs force the users to represent hierarchical data (or complex nested data or compound data) such as bill of materials in terms of tuples in multiple relations. This is awkward to start with. Further, to retrieve data thus spread out in multiple relations. RDBs must resort to joins, a general expensive operation. The data type of an attribute of an object in OOPLs may be a primitive type or an arbitrary user-defined type (class). The fact that an object may have an attribute whose value may be another object naturally leads to nested object representation, which in turn allows hierarchical data to be naturally (i.e., hierarchically) represented.

RDBs offers a set of primitive built-in data types for use as domains of columns of relation, but they do not offer any means of adding user-defined data types. The built-in data types are basically all numbers and short symbols. RDBs are not designed to allow new data types to be added and therefore often require major surgery to the system architecture and code to add any new data type. Adding a new data type to a database system means allowing its use as the data type of an attribute – that is, storage of data of that type, querying, and updating of such data. Object encapsulation in OOPLs does not impose any restriction on the types of data, that the data may be primitive types or user-defined types. Further, new data types may be created as new classes, possibly even as subclasses of existing classes, inheriting their attributes and methods.

Object encapsulation is the basis for the storage and management of programs as well as data in the database. RDBs now supports stored procedures – that is, they allow programs to be written in some procedural language and stored in the database for later loading and execution. However, the stored procedures in RDBs are not encapsulated with the data – that is, they are not associated with any linear relation or any tuple of a relation. Further, since RDBs do not have the inheritance mechanism, the stored procedures cannot automatically be reused.

Self-Assessment Questions - 6

20. The data type facilities are the keys to eliminating_____of the important deficiencies of RDBs.
21. Adding a new data type to a_____system means allowing its use as the data type of an attribute – that is, storage of data of that type, querying, and updating of such data.



8. DIFFERENCE BETWEEN RELATIONAL DATABASE MANAGEMENT SYSTEM AND OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM

RDBMSs were never designed to allow for the nested structure. These types of applications are extensively found in CAD/CAE, aerospace, etc. OODBMS can easily support these applications. Moreover, it is much easier and more natural to navigate through these complex structures in the form of objects that model the real world in OODBMS, rather than tables, tuples, and records in RDBMS.

It is hard to confuse a relational database with an object-oriented database. The normalized relational model is based on a fairly elegant mathematical theory. Relational databases derive a virtual structure at run time, based on values from sets of data stored in tables. Database constructs views of the data by selecting data from multiple tables and loading it into a single table (OODBs traverse the data from the object to object).

Relational databases have a limited number of simple, built-in data types, such as integer and string, and a limited number of built-in operations that can handle these data types. You can create complex data types in a relational database, but you must do it on a linear basis, such as combining fields into records. And the operations on these new complex types are restricted, again, to those defined for the basic types (as opposed to arbitrary data types or sub-classing with inheritance as found in OODBs).

The object model supports browsing of object class libraries, which allows the reuse, rather than the reinvention, of commonly used data elements. Objects in an OODB survive multiple sessions; they are persistent. If you delete an object stored in a relational database, other objects may be left with references to the deleted one and may now be incorrect. The integrity of the data thus becomes suspect and creates inconsistent versions.

In the relational database, complex objects must be broken up and stored in separate tables. This can only be done in a sequential procedure, with the next retrieval relying on the outcome of the previous. The relational database does not understand a global request, and thus cannot optimize multiple requests; OODBs can issue a single message (request) that contains multiple transactions.

The relational model, however, suffers at least one major disadvantage. It is difficult to express the semantics of complex objects with only a table model for data storage. Although relational databases are adequate for accounting or other typical transaction-processing applications, where the data types are simple and few in number, the relational model offers limited help when data types become numerous and complex.

Object-oriented databases are favored for applications where the relationships among elements in the database carry the key information. Relational databases are favored when the values of the database elements carry the key information. That is, object-oriented models, capture the structure of the data; relational models organize the data itself. If a record can be understood in isolation, then the relational database is probably suitable. If a record makes sense only in the context of other records, then an object-oriented database is more appropriate.

Engineering and technical applications were the first applications to require databases that handle complex data types and capture the structure of the data. Applications such as mechanical and electrical computer-aided design (MCAD and ECAD) have always used nontraditional forms of data, representing such phenomena as three-dimensional images and VLSI circuit designs. Currently, these application programs store their data in application-specific file structures. The data-intensiveness of these applications is not only in a large amount of data that needs to be programmed into the database but in the complexity of the data itself. In these design-based applications, relationships among elements in the database carry key information for the user. Functional requirements for complex cross-references, structural dependences, and version management all require a richer representation than what is provided by hierarchical or relational databases. The comparison of OODBMS and RDBMS is shown in table 13.1.

Table 13.1: Comparison of OODBMS with RDBMS

OODBMS	RDBMS
It emphasizes data encapsulation and data independence.	It emphasizes mainly data independence.
It stores methods and data.	It stores data only.
The data can be used only through their classes' methods.	The data can be partitioned on basis of users' requirements and on the specific users' applications.
Data and methods non-redundancy is achieved through encapsulation and inheritance. Inheritance helps to reduce the redundancy of methods.	Data normalization aims at reducing or eliminating data redundancy. It is used in the stage of designing the database and not in the stage of developing the applications.
Object-oriented databases are favored for applications where the relationships among elements in the database carry the key information.	Relational databases are favored when the values of the database elements carry the key information.
It follows object oriented properties.	It does not have object oriented properties.

Self-Assessment Questions - 7

22. Relational databases derive a _____ structure at run time based on values from sets of data stored in tables.
23. Objects in an OODB survive multiple sessions; they are _____.
24. In the relational database, complex objects must be broken up and stored in separate _____.
25. Object-oriented databases are favoured for applications where the relationships among elements in the database carry the information.
26. Relational databases are favoured when the values of the _____ elements carry the key information.

9. ALTERNATIVE OBJECT-ORIENTED DATABASE STRATEGIES

There are at least six approaches for incorporating object orientation capabilities in databases:

1. **Novel database data model/data language approach:** The most aggressive approach develops an entirely new database management system with object orientation capabilities. Most of the research projects in object-oriented databases have pursued this approach. The industry introduces novel DML (Data Manipulation Language) and DDL (Data Definition Language) constructs for a data model based on semantic and functional data models.
2. **Extending an existing database language with object-orientation capabilities:** A number of programming languages have been extended with object-oriented constructs. C++ flavors (an extension of LISP), and Object Pascal are examples of this approach in programming languages. It is conceivable to follow a similar strategy with database languages. Since SQL is a standard and the most popular database language, the most reasonable solution is to extend this language with object-oriented constructs, reflecting the object oriented capabilities of the underlying database management system. This approach is being pursued by most vendors of relational systems, as they evolve the next generation products. There have been many such attempts incorporating inheritance, function composition for nested entities, and even some support of encapsulation in an SQL framework.
3. **Extending an existing object-oriented programming language with database capabilities:** is to introduce database capabilities to an existing object-oriented language, object identity – will already be supported by the object-oriented language. The extensions will incorporate database features (querying, transaction support, persistence, and so on).
4. **Embedding object-oriented database language constructs in a host (conventional) language:** Database languages can be embedded in host programming languages. For example, SQL statements can be embedded in PL/SQL, FORTRAN, and Ada. The types of SQL (that is relations and rows in relations) are quite different from the type systems of these host languages. Some object-oriented databases have taken a similar approach with a host language and an object-oriented database language.

Self-Assessment Questions - 8

27. Most of the research projects in object-oriented databases have pursued _____ approach.
28. A number of programming languages have been extended with _____ constructs.
29. Database languages can be embedded in host _____ languages.



10. SUMMARY

During the past decade, object-oriented technology has found its way into database user interfaces, operating systems, programming languages, expert systems, and the like. In this unit, we discussed the advantages of OODBMS and the deficiencies of RDBMS. We discussed that OODB must provide standard database facilities found in today's relational database system (RDBS), including nonprocedural query facility for retrieving objects, automatic query optimization and processing, dynamic schema changes, automatic management of access methods to improve query processing performance, automatic transaction management, concurrency control, recovery from system crashes, and security and authorization.

Object-Oriented database product has already been in the market for several years, and several vendors of RDBMS are now declaring that they will extend their products with object-oriented capabilities. Nowadays, OODBMS is evolving in the industry to overcome all the drawbacks of RDBMS.

11. TERMINAL QUESTIONS

1. What are the drawbacks of current commercial databases?
2. What is the meaning of multi-media data?
3. List a few requirements for multi-media data management.
4. What are the two kinds of new data types supported in object-database systems? Give an example of each, and discuss how the example situation would be handled if only an RDBMS were available.
5. Compare RDBMSs with OODBMSs. Describe an application scenario for which you would choose an RDBMS, and explain why. Similarly, describe an application scenario for which you would choose an ORDBMS, and explain why.

12. ANSWERS

Self Assessment Questions

1. Object Oriented Database Management Systems
2. Knowledge-based Database Management Systems
3. Applications
4. Computer Oriented Software Engineering
5. Multimedia
6. knowledge base
7. Object
8. Encapsulation
9. Reuse
10. Method
11. class hierarchy
12. maintenance costs
13. data
14. reliable
15. database
16. increased security.
17. expert systems
18. hierarchical and relational
19. larger structures
20. Three
21. database
22. Virtual
23. Persistent
24. Tables
25. Key
26. Database
27. Novel database data model
28. object-oriented
29. programming

Terminal Questions

1. Existing commercial DBMS, both small and large have proven inadequate for the applications like computer-aided design (CAD), multimedia systems, software engineering, and knowledge database where operations require complex operations and data structure representation. (Refer section 1, 3 and 7 for detail)
2. Multi-media data are those data which includes text, numbers, images, graphics, and digital audio and video. Such multimedia data is typically stored as sequences of bytes with variable lengths, and segments of data are linked together for easy reference. (Refer section 3 for detail)
3. Multi-media data requires storage for the sequences of bytes with variable lengths, and segments of data that can be linked together for easy reference. The variable-length data structure cannot fit well into the relational framework, which mainly deals with fixed-format records. (Refer section 3)
4. In object oriented database systems, the data type of an attribute of an object in OOPLs may be (i) a primitive type (ii) an arbitrary user-defined type (class). (Refer section 7 for detail)
5. Both RDBMS and OODBMS are used to store and retrieve information to and from DBMS. But RDBMS does not support user-defined data types (class) and does not allow nested structure whereas OODBMS supports all of these and is more natural to navigate through the complex structures in the form of objects that model the real world in OODBMS, rather than a table, tuples, and records in RDBMS. (Refer section 8 for detail)