MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 12
Object Relational and Extended Relational Databases

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	3
	1.1 Objectives	-	-	
2	Object Relational Database	-	1, I	4-6
	2.1 Reasons behind the development of ORDBMS	-	-	
	2.2 Advantages of ORDBMS	-	-	
	2.3 Disadvantages of ORDBMS	-	-	
	2.4 Characteristics of object relational databases	-	-	
3	Extension Techniques in RDBMS	-	2	7-9
4	Standards for OODBMS Products and Applications	-	3	9-11
	4.1 ODMG-93 standards	-	-	
	4.2 ODMG Smalltalk binding SQL3	-	-	
5	Nested Relations and Collections	1	4	12-14
6	Storage and Access Methods	-	5	15-16
7	Implementation Issues for Extended Type	-	6	17
8	Comparing RDBMS, OODBMS and ORDBMS	1	7, II	18-19
9	Summary	-	-	20
10	Glossary	-	-	21
11	Terminal Questions	-	-	21
12	Answers	-	-	22-23
13	References	-	-	23

1. INTRODUCTION

In previous units, you have read about databases in general with an emphasis on relational databases. Relational databases, though quite popular, have some demerits, especially when it comes to the question of representing real-world entities.

There are some other database models worth mentioning, like object oriented databases and object relational databases which have advantages over relational databases because of their object based approach. In these models, information is represented in the form of objects as used in object- oriented programming. Object databases main utilisation is in object oriented areas.

In this unit, you will study the features of these two objects relational and extended relational databases. You will learn about the general practices and approaches associated with these two models. Through this, we will provide a basic structure for understanding object-oriented database management system. We will focus primarily on design techniques used in RDBMS, extension techniques in RDBMS, standards for OODBMS products and applications. We will also discuss nested relations and collections, storage and access methods and Implementation issues for extended type. We will conclude with a comparison of RDBMS, OODBMS and ORDBMS.

1.1 Objectives

After studying this unit, you should be able to:

- *Describe object-relational DBMSs*
- *State the design techniques used in RDBMS*
- *Discuss the extension techniques in RDBMS*
- *Identify the OODBMS standards*
- *Recognise the storage and access methods in ORDBMS*
- *Analyse the implementation issues for extended types*
- *Differentiate between RDBMS, OODBMS and RDBMS*

2. OBJECT RELATIONAL DATABASE

Let us start out discussion with the basic concepts of Object Relational database. This section 12.2 will lay the foundation for better understanding of the subsequent topics.

Database systems that are based on the object relational model are known as Object relational databases. In simple words, object relational database are the 'Relational Database **+** Object Oriented Features'. For the relational database users, it is easy to migrate to object relational databases.

Object relational data models extend the relational data model by providing a richer type system including object orientation and add constructs to relational query languages, such as SQL to deal with added data types.

Some examples of databases based on Object Relational technology are:

- Informix Dynamic Server (formerly Postgres, Illustra and Informix Universal Server)
- IBM's DB2 Universal Database
- Oracle-8
- /X, OSMOS
- Ingres II
- Sybase Adaptive Server

Most of the above databases are the enhancement of their relational predecessors. They provide full support for multimedia and web, various types of ADT (Abstract data types), spatial data and geographic data, time series data, collection valued attributes, and others. Thus, they are well equipped with facilities for efficient application development.

2.1 Reasons behind the development of ORDBMS

In the commercial world, there are many DBMS available. Then you may wonder what the reasons behind the development of ORDBMS are. We will briefly discuss the main force behind development of ORDBMSs.

ORDBMS basically, evolved to meet the challenges of new applications. New applications are complex and sophisticated and have diverse data needs. They require various types of data such as text, images, audio, Streamed data and BLOBs (binary large objects) etc to record them into the system.

In addition, there is rising trend to amalgamate the best features of object databases into the relational model so that the developers can meet the growing challenges of developing the new applications.

All these factors led to the development of the object relational database, which we see in the market.

2.2 Advantages of ORDBMS

ORDBMS are widely used now-a-days because of the various advantage linked to it such as;

- The main advantage of object relational data model arises from the concept of "reuse and sharing". By reuse, we mean that the programmer can easily extend the DBMS server to achieve the standard functionality centrally, rather than coding it in each and every application.
- Sharing means that if the developer wants the applications to use particular database functionality, then it can be embedded in the functionality of the server.
- ORDBMS allows enterprises to easily migrate from their existing systems to an ORDBMS without making major changes.
- In addition, a user may easily make use of object-oriented systems in parallel of the RDBMS features.

2.3 Disadvantages of ORDBMS

In spite of various advantages and benefits, ORDBMS still has some disadvantages. Some of these are listed below:

1. The ORDBMSs approach is complex in nature and is associated with increased costs.
2. The Relational database proponents have a view point that the extensions to relational model in the form of ORDBMS have diluted the simplicity of relational model which was the major factor behind its success.
3. Some database experts also believe that the ORDMSs will be of use for only a limited set of applications which may not be practical for relational technology.

4. Also the physical architecture of object-relational model is not suitable for handling high-speed web applications.

2.4 Characteristics of Object Relational Databases

Some of the important characteristics of object relational databases are as follows:

- Nested relations
- Complex types and object-orientation
- Querying with complex types
- Creation of complex values and objects

We will discuss about these features in detail in the coming sections.

SELF-ASSESSMENT QUESTIONS – 1

1. For the relational database users, it is easy to migrate to object relational databases. (True/False)
2. The main advantage of object relational data model arises from the concept of "________________"

Activity 1

With the help of internet, find a few examples of object relational database in real-life. For example, Oracle has recently introduced Oracle8, an ORDBMS.

3. EXTENSION TECHNIQUES IN RDBMS

Commercial RDBMS can be extended to incorporate the features of object oriented languages. This concept of extensibility is the basic idea behind the innovation of ORDBMS technology.

As you studied in section 12.2, it is very difficult to model complex data structures such as Streamed data, BLOB, CLOB etc. in a RDBMS. Hence, the developers proposed some ideas to solve this problem. One idea to handle this problem is that the DBMS vendors must provide more data types and functions built into their products itself.

But this approach is not practical, as there exists various types of new data types. Hence a better solution was proposed that suggested to build the DBMS engine that is capable of accepting the addition of new, application specific functionality.

Therefore the database developers can now specialise or extend many of the features of an ORDBMS such as the data types, OR-SQL expressions, the aggregates, and so on. ORDBMS acts as a framework into which specialised software modules can be embedded.

Extensions to a RDBMS mainly fall into following categories:

- Use of Type Constructors to denote complex objects
- Object Identifiers using references
- Support for additional or extensible data types
- Support for user-defined routines (procedures or functions)
- Unstructured complex objects Now, let us discuss it in brief.

Type Constructors: The type constructors are used to specify complex types. As the user defines them they are also well-known as user-defined types (UDTs).

A row type is declared using the following syntax:

CREATE TYPE row_type_name **AS [ROW]** «component declarations»; An array type is used to define a collection. For example:

CREATE TYPE Cust-type AS ();

Custname VARCHAR (20),

Contactnumber INTEGER Array [5]

Object Identifiers Using References: A user-defined type can be used in two ways: either as type for an attribute or to define the row types of tables. For example, two tables can be created based on the row type declarations as follows:

```
CREATE TABLE Employee OF Emp_.type REF IS emp_id SYSTEM GENERATED;
CREATE TABLE Company OF Comp_type
REF IS comp_id SYSTEM GENERATED,
PRIMARY KEY (compname));
```

We can openly declare the object identifiers in the type description rather than in the table declaration.

Support for additional or extensible data types: SQL-92 specifies the syntax and behaviour for about ten data types. SQL-3, consist of about a hundred more of data types which include geographic, temporal, text types, etc.

Many ORDBMS provide easy support for these common extensions by means of commercial packages of Abstract data types (ADTs) products.

For example Data Cartridge is an application package for Oracle, Data Blades is for Informix Universal Server and Extenders in DB2. Yet to rely on the DBMS trader to deliver all of the innovation data types extension does not fully deal with the problem.

Therefore with an object-relational DBMS, developers can implement data types of their own depending upon the application. Developers also define the actions associated with those data types.

Support for user-defined routines (procedures or functions): RDBMS provides integral functions for user defined types(UDT). For example, you had a UDT called Type_T. Now a constructor function Type_T (), will return a new object of that type. This new object's, each quality will be initialised to its default value.

Unstructured complex objects: Some more data types for binary large objects (BLOBs), and large object locators are extended from RDBMS. Binary large objects i.e. (BLOBs) and character large objects (CLOBs) are the two variations exist.

SELF-ASSESSMENT QUESTIONS – 2

3. ORDBMS acts as a framework into which specialised software modules can be embedded. (True/False)
4. Which of the following is the application package for Informix Universal Server?
   a) Extenders
   b) Data blades
   c) Data cartridge
   d) Blade smith

4. STANDARDS FOR OODBMS PRODUCTS AND APPLICATIONS

Object Data Management Group (ODMG) represents an association of ODBMS vendors. It was formulated to provide a standard for various ODBMS products.

The objectives of ODMG association are as follows:

- To persist to develop the ODMG standard,
- To train the developer community regarding the standard,
- To advance its use, and
- To offer official recognition of conformity to the standard.

4.1 ODMG-93 Standards

A working group composed of five representatives of the OODBMS vendors established ODMG-93 standard. It provides an inclusive framework for designing an object database, writing portable applications in C++ or Smalltalk, and querying the database with a simple and very powerful query language.

Major components of ODMG-93 standard: The ODMG-93 standard consists of the following parts:

- ***Object model:*** It is the model on which the fundamental concepts of object-oriented data structures such as classes, objects, encapsulation, inheritance, interfaces, operations, etc. depend upon.

- ***Object definition language (ODL):*** Object definition language defines how to formulate a database schema (the structure of a database). It is neutral to programming languages.
- ***Object interchange format:*** This determines how the various objects are represented for exchanging them between different OODBMS.
- ***Object query language (OQL):*** OQL is the query language for ODMG object model. Thus, it is projected to retrieve data from an object base. It is not concerned with keeping it up to date and neither has it explained the SQL-like abstractions such views, constraints and stored procedures. It is intended to work directly with various programming language for which a requisite is defined such as C++, Java or SMALLTALK.
- ***Bindings to programming languages C++, Smalltalk and Java***: It is the ODMG bindings to programming languages like C++, Java and Smalltalk.

4.2 ODMG Smalltalk Binding

The ability to store, retrieve and modify persistent objects in Smalltalk is provided by the ODMG Smalltalk binding. A mechanism is provided by the binding so as to invoke OQL and measures for transactions and operations on databases. Reachability makes Smalltalk objects persistent. Thus, an object turns into persistent when another persistent object in the database references it, and when it is no longer reachable, it is garbage-collected.

All of the classes described in the ODMG object model are directly mapped to Smalltalk classes by the Smalltalk binding. Wherever possible, the Object Model collection classes and operations are mapped to standard collection classes and methods. Relationships, transactions and database operations are also mapped to Smalltalk constructs.

4.3 SQL3

ANSI and ISO have developed a new SQL standard as SQL3. It is assumed to be a programming language with full computational and pragmatic power.

The main features of SQL-3 are listed below:

- Various types of user-defined abstract data types (ADTs) are supported by SQL-3. It also supports various methods, object identifiers, subtypes, inheritance and polymorphism.
- The statements defining tables are extended by SQL, especially, the types of rows, row identifiers, and (specific) inheritance between rows of families of tables.
- Each SQL3 table has a predefined column called as IDENTITY that contains row identifiers which can be used as values in other rows.

Thus, through SQL3, it is likely to create pointer-based, network data structures in the style of DBTG CODASYL.

- The ADT (abstract data types) theory in SQL3 helps the database developer to correlate values stored in the tables with methods. In such a case, these values are not directly within reach, but entirely by methods.
- Methods can be indicated in SQL3 or other languages such as C++ and Java.
- SQL3 initiated a lot of extensions to earlier SQL versions, including e.g. enriched support for BLOBs, CLOBs, collections, triggers, transactions, cursor processing, etc.
- It also comprises some new features such as support for user-defined aggregate operations, transitive closures, and even deductive rules.

SELF-ASSESSMENT QUESTIONS – 3

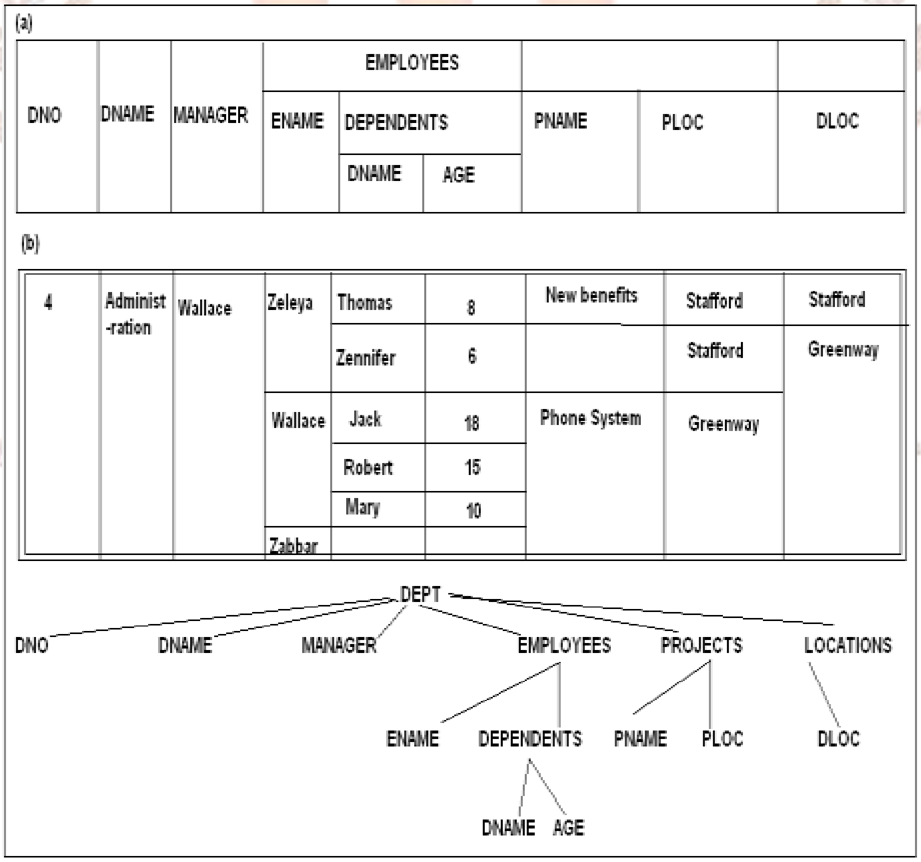
5. _________________ language defines how to formulate a database schema (the structure of a database).
6. The ____________________ concept in SQL3 helps the database developer to associate values stored in the tables with methods.
7. Select the name of the predefined column in SQL 3 that contains row identifiers which can be used as values in other rows.
   a) ADT
   b) Key
   c) Identity
   d) Index

5. NESTED RELATIONS AND COLLECTIONS

All the restrictions and constraints of first normal form are removed from the nested relational model from the basic relational model. It is also called Non- lNF or Non-First Normal Form (NFNF) or NF2 relational model. It is required that the characteristic of the basic relational model (also called the flat relational model) to be single-valued and to have atomic domains.

Composite and multivalued features, which results into complex tuples that have a hierarchical arrangement, are permitted by the nested relations. Objects that are naturally hierarchically ordered are characterised by nested relations.

In Figure 12.1, part (a) shows a nested relation schema DEPT based on part of the COMPANY database, part (b) gives an example of a Non-INF tuple in DEPT and (c) tree structure of relation schema.



(a)

			EMPLOYEES					
DNO	DNAME	MANAGER	ENAME	DEPENDENTS		PNAME	PLOC	DLOC
				DNAME	AGE			

(b)

4	Administ -ration	Wallace	Zeleya	Thomas	8	New benefits	Stafford	Stafford
				Zennifer	6		Stafford	Greenway
			Wallace	Jack	18	Phone System	Greenway	
				Robert	15			
				Mary	10			
			Zabbar					

Fig 12.1: (a) A Nested Relation Schema DEPT (b) An Example of a Non-INf tuple in DEPT (c) Tree Structure of Relation Schema

In order to classify the DEPT schema in the form of a nested structure, the following can be written:
dept = (dno, dname, manager, employees, projects, locations) employees = (ename, dependents) projects = (pname, ploc) locations = (dloc) dependents = (dname, age)

- Define all attributes of the DEPT relation.
- Then the nested attributes of DEPT viz. EMPLOYEES, PROJECTS, and LOCATIONS are themselves defined.

The second-level nested attributes, viz. DEPENDENTS of EMPLOYEES, are defined, in the next step and so on. In the nested relation definition, the names of the various attributes must be dissimilar for each of them.

A nested attribute is usually a multivalued composite attribute and within each tuple, it leads to a "nested relation". For example, a relation is defined with two attributes (PNAME, PLOC) for the value of the PROJECTS attribute within each DEPT tuple.

The PROJECTS attribute in the DEPT tuple of Figure 12.1 (b), contains three tuples as its value. Attributes, such as LOCATIONS of DEPT, may be multivalued simple attributes.

Generally, nested relational models treat an attribute to be multivalued, whereas it is possible to have a nested attribute that is single-valued and composite.

A nested relational database schema consists of various external relation schemas that define the top level of the individual nested relations. Nested attributes define relational structures that are nested inside another relation, thus, they are also called internal relation schemas. Here in our case, DEPT is the only external relation.

On the contrary, EMPLOYEES, PROJECTS, LOCATIONS, and DEPENDENTs are internal relations. At last, the attributes that are not nested are simple attributes and they appear at the leaf level.

Each relation schema by means of a tree structure can be represented as shown in Figure 12.1 (c). In this tree structure, the root represents an external relation schema, the leaves symbolise simple attributes, and the internal nodes are internal relation schemas.

The three first-level nested relations in DEPT characterise independent information. Therefore, EMPLOYEES correspond to the employees working for the department, PROJECTS are the projects controlled by the department, and LOCATIONS symbolises the diverse department locations.

The schema does not represent the relationship between EMPLOYEES and PROJECTS; this being an M: N relationship is difficult to represent in a hierarchical arrangement.

Extensions to the relational algebra, the relational calculus and SQL, have been projected for nested relations. Conclusively, nested collections propose immense modelling power but also bring up difficult design decisions.

SELF-ASSESSMENT QUESTIONS – 4

8. Nested relational model is also known as ___________________.
9. Nested attributes are also called internal relation schemas. (True/False)

6. STORAGE AND ACCESS METHODS

ORDBMS data storage is quite similar to an RDBMS. In this the data is organised into pages. A collection of pages holds all the data for a respective table. There is also the facility of caching wherein the blocks of memory are retained by the ORDBMS particularly for this purpose to store repeatedly visited pages.

Caching avoids the expenditure of referring to disk every time the page is accessed. As the user generates a query, the system retrieves the pages from disk and expel out from the memory cache. If the cache is full, then the altered pages, or occasionally visited pages, may perhaps be stored back to disk. The ORDBMS passes pointers to the memory occupant data as opinions.

The ORDBMS is least bothered regarding the function it performs, or the way it performs it. It is only concerned in relation to the logic's return value. The ORDBMS has the right to read more data from pages in memory, every time a single execution of the embedded logic is completed. It is also liberated to raise the logic again with other point of views.

The ORDBMS do not store the logic with the object data; rather it combines them jointly at run-time.

Large object data storage: Large object data has precise set of disputes. As extensible database applications are expected to involve a lot of large data objects, dealing with them proficiently is principally important.

The table records use relatively small data pages, usually only a few kilobytes in size. It would be an unacceptable constraint to require the new types to fit within these pages, so identifications (IDs) offers exceptional means for supporting data objects of any size.

A considerable increase in the size of the table results from storing large object data like the polygon consequent to a state boundary, with table data, such as the city's name and present population. If it is believed that less queries access the large object data, then a tactic like this would significantly increase the time consumed to complete a major percentage of queries.

The storage of large object data from the table data is separated by the ORDBMS. The handle for the large object data is stored with the row if the value of a data object is stored as a large object only. This handle is used by the user-defined functions that implement behaviour for large objects to open the large object and access its contents.

Indexing new types: One important reason for users to place their data in a database is to allow for efficient access via indexes. Unfortunately, the standard RDBMS index structures support only equality conditions (B+ trees and hash indexes) and range conditions (B+ trees). An important issue for ORDBMSs is to provide efficient indexes for ADT methods and operators on structured objects.

One way to make the set of index structures extensible is to publish an *access method interface* that lets users implement an index structure outside of the DBMS. The index and data can be stored in a _le system, and the DBMS simply issues the *open, next,* and *close* iterator requests to the user's external index code.

Such functionality makes it possible for a user to connect a DBMS to a Web search engine, for example. A main drawback of this approach is that data in an external index is not protected by the DBMS's support for concurrency and recovery.

An alternative is for the ORDBMS to provide a generic `template' index structure that is sufficiently general to encompass most index structures that users might create. Because such a structure is implemented within the DBMS, it can support high concurrency and recovery.

The *Generalised Search Tree* (GiST) is such a structure. It is a template index structure based on B+ trees, which allows most of the tree index structures invented so far to be implemented with only a few lines of user- defined ADT code.

SELF-ASSESSMENT QUESTIONS – 5

10. ORDBMS does not care about the logic's return value. (True/False)
11. The standard RDBMS index structures support only equality conditions (B+ trees and hash indexes) and range conditions (B+ trees). (True/False)

7. IMPLEMENTATION ISSUES FOR EXTENDED TYPE

The extended type systems with related functions (operations) have a variety of execution issues. These are briefly described below:

- The ORDBMS must link a user-defined function in its address space in a dynamic manner only when it is required. Dynamic linking is available in various types of ORDBMS. The DBMS address space is increased by a stagnant connections of the functions by an order of scale.
- Client-Server concerns relate with the activation and placement of the functions. There is a large overhead involved in doing it remotely hence it must be done in the DBMS address space.
- It must be possible to execute the queries inside functions in DBMS. A function must be capable of operating in a similar manner irrespective of the fact that whether it is being executed from an API or by DBMS as a part of SQL statement.
- The DBMS must provide efficient storage and access of data as ORDBMS supports a variety of data types and operators associated with it. Especially given new types, is very important for an ORDBMS.
- There exist several other issues relating to Object-relational database design which are important to be addressed. Some of these are given below:
  - ❖ Object-relational design is more complicated
  - ❖ Query processing and optimisation
  - ❖ Interaction of rules with transactions

SELF-ASSESSMENT QUESTIONS – 6

12. Which linking is available in various types of ORDBMS?
    a) Dynamic linking
    b) Refresh linking
    c) Static linking
    d) Pure linking
13. It must be possible to execute the queries inside functions in DBMS. (True/False)

8. COMPARING RDBMS, OODBMS AND ORDBMS

After reading so much about the various types of database systems, you can now understand the difference between them and compare them. Hence, in this section we will compare the three important types of database systems i.e. RDBMS, OODBMS and ORDBMS. Table 12.1 gives the comparison between three database systems.

Table 12.1: Comparison between RDBMS, OODBMS and ORDBMS.

Criteria	**RDBMS**	**ODBMS**	**ORDBMS**
Defining standard	SQL2	ODMG-2.0	SQL3 (in process)
Support for object-oriented features	No support provided; Program object is hard to be mapped to the database	Extensive Support provided	Restricted support; mainly to new data type
Usage	Uncomplicated usage	Reasonable for programmers; some SQL access for end users	Easy to use except for some extension
Support for complex relationships	Abstract datatypes are not supported	An extensive range of datatypes and data with complex inter-relationships is supported	Abstract datatypes and complicated relationships
Performance	High	Moderate	Expectations to perform very well
Product maturity	Comparatively developed and so very mature	Concept is not many years old, and so relatively mature feature	Still developing, so immature
SQL Usage	Extensively supports SQL	OQL is similar to SQL, but with additional features like Complex objects and object-oriented features	SQL3 is being developed with OO features integrated in it

Advantages	Its dependence on SQL, relatively simple query optimisation hence good performance	Complex applications can be handled, reusability of code, less coding	It can query compound applications and is capable of handling large and complex applications
Disadvantage	It is unable to deal with complex applications	Low performance due to difficult query optimisation, it is unable to support large- scale systems	Cannot perform well in web application
Support from vendors	It is treated as highly successful so it has a huge market size but many suppliers are shifting to ORDBMS	It lacks supplier support due to the enormous size of RDBMS market	Very good future among RDBMS suppliers

SELF-ASSESSMENT QUESTIONS – 7

14. Out of RDBMS, OODBMS and ORDBMS, which has got the best performance?

15. Which one of the following is the easiest to use?

    a) ORDBMS

    b) OODBMS

    c) RDBMS

    d) DBMS

Activity 2

Prepare a comparative study with an example of each between RDBMS, OODBMS and ORDBS.

9. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- An ORDBMS is a data repository that can be extended to manage any kind of data and to organise any kind of data processing. It represents the next logical step in the evolution of DBMS technology. It both preserves those aspects of relational DBMS technology that proved so successful (abstraction, parallelism, and transaction management) and augments the features with ideas, such as extensibility and advanced data modelling, derived from object-oriented approaches to software engineering,
- Users can also implement their own extensions as needed by using the ADT facilities of these systems. We briefly discussed some implementation issues for ADTs. Finally, we gave an overview of the nested relational model, which extends the flat relational model with hierarchically structured complex objects.
- ODMG 3.0 was developed by the Object Data Management Group (ODMG). The ODMG is a consortium of vendors and interested parties that work on specifications for object database and object-relational mapping products.
- The major components of ODMG 3.0 specification are Object Model, Object Definition Language (ODL), Object Query Language, • Object Interchange Format, C++, Java and Smalltalk Language Binding.
- ODMG Smalltalk binding is the binding of ODMG implementations to Smalltalk.
- There are various implementation issues regarding the support of extended type systems with associated functions (operations).

10. GLOSSARY

- ***Abstract data type:*** A combination of an atomic data type and its associated methods is called an abstract data type, or ADT.
- ***Access Method:*** The method used to store, find and retrieve the data from a database.
- ***BLOB:*** Binary Large Object. Generally used to store multimedia data such as video, images and audio. It stores data in binary format only
- ***CLOB:*** Character Large Object. It is used for documents or large strings that use the database character. It stores this string data in the database character set format. NCLOB: National Character large object. Fixed-width multibyte CLOB. It is used for documents or large strings that use the National Character Set. It stores this string data in the National Character Set format.
- ***ODMG Smalltalk binding:*** This is the binding of ODMG implementations to Smalltalk.

11. TERMINAL QUESTIONS

1. What do you mean by object relational database? Also explain its advantages and disadvantages.
2. Explain the characteristics of object-relational databases.
3. Write short notes on ODMG standards.
4. Briefly describe SQL3 and its important features.
5. What are the main reasons behind the development of ORDBMS
6. Describe the various implementation issues for extended types.
7. Differentiate between RDBMS, OODBMS and ORDBMS
8. What are the various components of ODMG-93.
9. What are the storage methods in ORDBMS? Briefly describe.

12. ANSWERS

Self-Assessment Questions

1. True
2. reuse and sharing
3. True
4. b (Data Blades)
5. Object definition
6. ADT(abstract data types)
7. C (Identity)
8. Non-First Normal Form (NFNF)
9. True
10. False
11. True
12. a (Dynamic linking)
13. True
14. RDBMS
15. C (RDBMS)

Terminal Questions

1. Database systems that are based on the object relational model are known as Object relational databases. Refer Section 2 for more details.
2. Some of the important characteristics of object relational databases are Nested relations, Complex types and object-orientation, Querying with complex types etc. Refer Section 2 for more details.
3. ODMG-93 provides specifications for object database and object- relational mapping products. Refer Section 4 for more details
4. SQL3 is a new SQL standard developed by ANSI and ISO. Refer Section 4 for more details.
5. ORDBMS evolved to meet the challenges of new applications. Refer Section 1 for more details.

6. ORDBMS implementation issues for extended types include various Client-Server issues, storage and access issues etc. Refer Section 6 for more details.
7. There are various differences between a RDBMS, ODBMS and ORDBMS. These have been discussed in table 12.1 in section 12.7 for more details.
8. The major components of ODMG 3.0 specification are Object Model, Object Definition Language (ODL), Object Query Language, Object Interchange Format, C++, Java and Smalltalk Language Binding. Refer Section 4 for more details.
9. In ORDBMS data is organised into pages. As similar to RDBMS storage. Refer Section 5 for more details.

13. REFERENCES

References

- Peter Rob, Carlos Coronel, (2007) *"Database Systems: Design, Implementation, and Management",* (7th Ed.), India: Thomson Learning.
- Silberschatz, Korth, Sudarshan, (2010) "*Database System Concepts",* (6th Ed.), India: McGraw-Hill.
- Elmasari Navathe, (1989) "*Fundamentals of Database Systems*", (3rd Ed.), Pearson Education Asia.

E-references

- http://infolab.usc.edu/
- http://www.odbms.org/
- infolab.stanford.edu/
- db.cs.berkeley.edu/