BLOCK III:
INTRODUCTION TO OBJECT ORIENTED, DISTRIBUTED, MULTIMEDIA AND SPATIAL DATABASES

UNIT 1: OBJECT ORIENTED DATABASE SYSTEM

Space for learners:

Unit Structure:

1.1 INTRODUCTION

In the earlier chapters, learners have been acquainted with some advanced form of traditional database concepts. In this chapter, the learners are going to be acquainted with a new dimension or extension area of the existing traditional database theory. All the data are imagined or visualized as some objects in this new area of discussion in

addition to the relational form of existing database management system. The data here are stored, manipulated and accessed as objects, which is done in object-oriented programming paradigms. The concept of object can be realized by defining one class with underlying characteristics like data abstraction, information hiding, encapsulation and imposing on it other object-oriented features like inheritance, polymorphism, early binding and late binding. The idea of object-oriented database approach comes into existence because of the acceptance of object-oriented programming approach among wide range of users worldwide. Some object databases, accepted widely and appreciated by the database community are mentioned in this unit. The required standards in the design of the object-oriented database systems and the associating query languages needs to be discussed in order to have a detailed insight into it. The relational database system is the basis on which the OORDBMS approach is evolving. The history of object relational database system is covered, which is followed up by its detailed description. The object-oriented relational database management approach is compared with the classic relational database management approach as the conclusive topic.

Space for learners:

1.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- Learn the object-oriented database system.
- Lean the need of object-oriented approach in databases.
- Learn the advantages and disadvantages of object-oriented databases.
- Accustom with some popular object databases.
- Learn the standards, languages and design issues of object based databases.
- Understand the history of object-oriented relational database system.
- Understand how object-oriented relational database system works.
- Compare RDBMS and OORDBMS.

1.3 CONCEPTS OF OBJECT-ORIENTED DATABASES

Space for learners:

A database is generally considered as data storage. This storage is further used for the purpose of searching, editing or updating, generating reports etc. Data storages can further be classified in four widely spelled categories viz., Traditional File System, Relational DBMS, Object Oriented DBMS and Object-Oriented Relational DBMS. These categories are classified on looking into the pattern of data.

The object-oriented paradigm is the basis upon which the object-oriented database is designed. The data or the information in the object-oriented database is represented and stored in the form of certain objects. The object-oriented database is also known as object database and is handled though object-oriented database management systems (OODBMS).

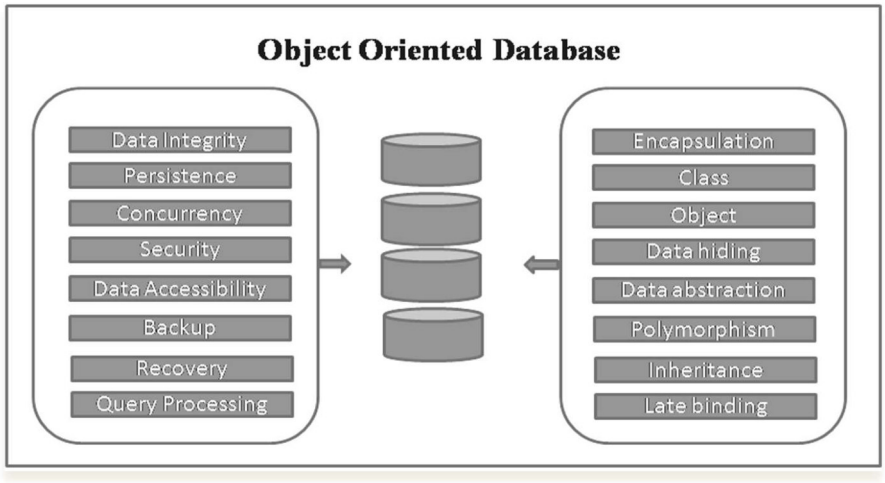


Fig-1.1: Conceptual bock diagram for OODBMS

The OODBMS encompasses the conventional DBMS features as well as the object-oriented features together. The conventional DBMS features are like data integrity, persistence, concurrency, security, backup, recovery query processing etc., while the object-oriented features are encapsulation, class, object, overloading, overriding, inheritance, early binding, late binding etc. Some of the popularly known object-oriented programming (OOP) languages are C++, Java, Perl, Ruby, Python and Java-script. Object-oriented databases are administered through the object database management systems

(ODBMS). The preparatory idea of object-oriented databases immerged in the late nineties of the nineteenth century and currently it has become common for various OOP based languages, such as C++, Java, Smalltalk and LISP. For example, Smalltalk is used in GemStone, LISP is used in Gbase, and COP is used in Vbase and so on.

Objects are composed of some data members and member functions or methods, which are encapsulated within a single unit with individual values and certain properties. Objects come into existence by instantiation of certain user defined classes. Objects generally go through a cycle that includes the creation or allocation of objects, use of the objects and the deletion or de-allocation of objects. Object databases are common among many modern high performances applications with high speed data access and manipulative facilities. Some of the significant areas where object databases are taking a pivotal role are the real-time systems, architectural engineering for 3-D modeling, telecommunications, robotics, molecular science, astronomy and many more.

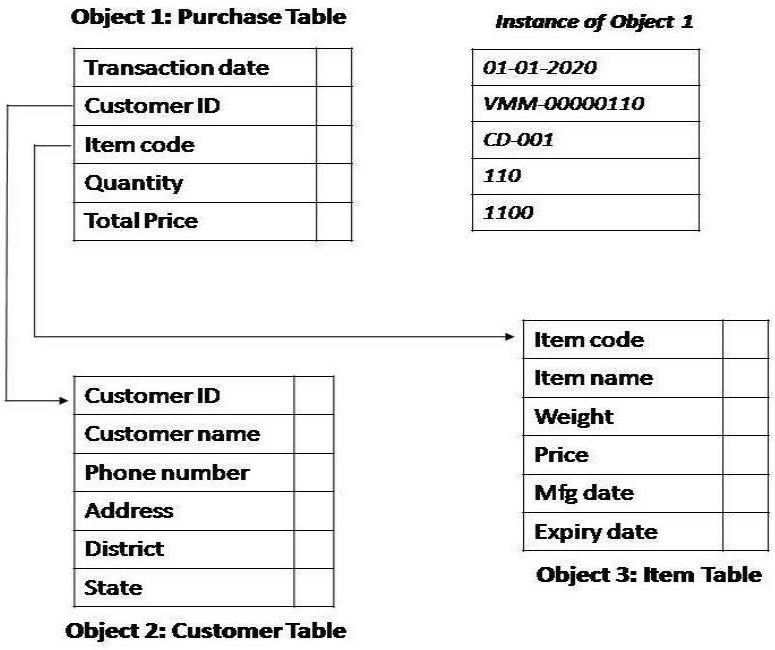


Fig-1.2: Object Oriented Model in OODBMS

Space for learners:

1.3.1 Key Features of Object Databases

- Object oriented databases or Object-oriented DBMS systems provide persistent storage to objects.
- It is capable of storing and reading data and converting the same into program objects for further storing of reading data & loading object based data in memory.
- Object databases bring permanent persistence to objects.
- The reading and mapping of the data of an object database to the objects is direct without any API like tool.

Object databases facilitate quick access of data or information and better performance inevitably. There are some object based databases with multi-lingual supports too. For example, Gemstone is such an object database that supports C++, Smalltalk and Java programming languages.

1.3.2 Drawbacks of Object Databases

- Object databases are still not popular among vast community of database users as compared to RDBMS.
- Developers are less in numbers in handling of object databases.
- Not many programming language support object databases.
- RDBMS have SQL as a standard query language. Object databases do not have such a standard.
- Object databases are difficult to learn for non-programmers

1.3.3 Popular Object Databases

Following are some of the popular object databases. These databases are accepted by most database users because of the highly flexible features that conform to the needs of current users. The descriptions of few such databases are mentioned below.

Space for learners:

Cache

Cache is developed by Inter Systems and it is a high-performing object database. This object based database facilitates a set of services that include data storage, concurrency management and handles diverse transactions issues and process management activities. Cache engine can be treated as full-fledged powerful database toolkit with extensive relational database features. This database can be used for diverse queries and modification purposes using standard SQL via ODBC, JDBC or object based methods. The computational efficiency of Cache is enormous and it is a most reliable relational database with high scalability parameters. Some of the important features of Cache database are mentioned below.

- Able to model data as objects, while eliminating mismatch between databases and object-oriented applications.
- Supports user-defined data types.
- The ability to take the advantage of methods and inheritance like functions.
- Object-extensions for SQL to handle object identity and relationships.
- The ability to avail SQL and object-based access through a single application.
- Clustering is used to store data ensuring maximum performance.

ConceptBase

Concept Base is another database system with multi-user and object-oriented support which is deductive in nature. It is a powerful tool for meta-modeling and is very useful for customizing modeling languages. Concept Base comes with an associating graphical user interface (GUI) facilitating the users with some common routines. Concept Base is developed by the Concept Base Team at University of Skövde (HIS) and the University of Aachen (RWTH). Commonly available operating systems like Linux, Windows and Mac support Concept Base. There is also a pre-configured virtual application within Concep Base, which contains associating executable files and source files along with the

Space for learners:

tools for compiling. The system is distributed under a FreeBSD-style license.

ObjectDB

Object DB is a powerful object-oriented database management system (ODBMS) based on Java language. It is a compact but reliable system, which is easy to use and extremely fast in terms of object database access. It supports both the client-server mode and the embedded mode. Object DB provides all the standard database management services. This is the reason, why the development process gets easier and the applications behave faster. It is capable of handling advanced level queries and providing enhanced indexing facilities. It is very much effective in multi-user environments, where there is always a rush of users. ObjectDB can easily be embedded in any applications irrespective of its sizes and types. This is such a database, which has been tested with Tomcat, Jetty, GlassFish, JBoss and Spring.

Several other popular object based databases are ObjectDatabase++, GemStone/S, Perst, ZODB, Wakanda, ODABA, Objectivity/DB. The discussions on these object databases are beyond the scope of this syllabus. The learners can use various internet sources to gather a detailed knowledge on these object based databases.

1.4 STANDARDS, LANGUAGES AND DESIGN

There should always be a standard agreed upon by all vendors of a particular type of database system. A standard can be resembled with an agreed roadmap maintaining uniformity among all stakeholders to proceed through a common model.

1.4.1 Standards and Languages

Some of the sound reasons for the need of standards are as follows.

- Standard provides support in maintaining the portability of database applications. Portability is defined as the capability to execute particular software or application on different platforms with minimal modifications.

Space for learners:

- Standards help in achieving interoperability. Interoperability refers to the ability of an application to access multiple systems. Here, the same application program may access some data stored under one ODBMS package, and another data stored under another source or package.
- Standard allows customers to compare commercial products of various vendors more easily by determining which parts of the standards are applied in their purchased product.

ODMG (Object Data Management Group) is an association for monitoring the object-oriented database management activities. This association proposed a standard for ODBMS in the year 1993 and it was named as ODMG 1.0 followed by ODMG 2.0 in 1995 and ODMG 3.0 in 2000.

The ODMG 3.0 standard has the following major specifications:

- Object Model
- Object Definition Language (ODL)
- Object Query Language (OQL)
- C++ Language Binding
- Smalltalk Language Binding
- Java Language Binding

1.4.1.1 Object Model

The object model specifies the ODMS based constructs. The basic building blocks of the object model are – *objects* and *literals*.

An object is referred to as the instance of its class type. The *state of an object* is composed of the values that the object carries for a certain set of properties. On the other hand, *the behavior of an object* is defined by the set of operations executed by the objects.

Space for learners:

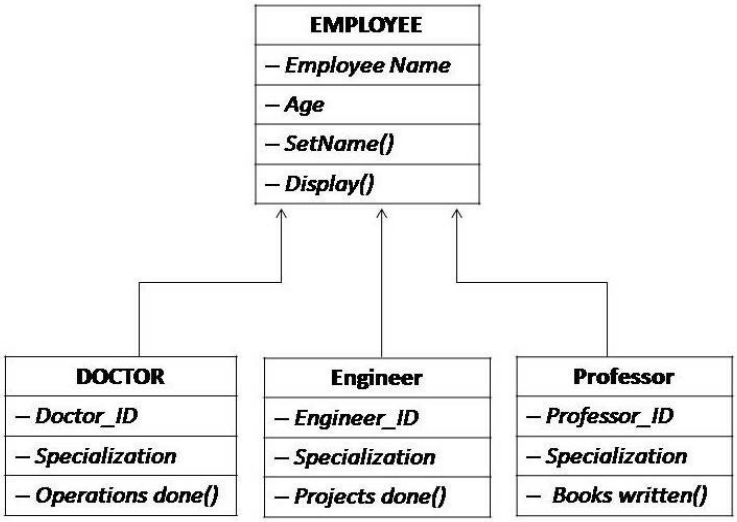


Fig-1.3: Hierarchy of classes/objects in OODBMS

An object is described with some associating parameters i.e. identifier, name, lifetime and structure. The details of these parameters are mentioned below.

Object Identifier: An object can be differentiated from all other nearby objects within its storage domain by using the object identifier. The objects always preserve the same object identifier in its lifetime during execution of a computer program. Thus, the value of an object's identifier never changes. The object remains the same, even if its attributes or the relationship values change. Object identifiers are generated by the OODBMS, but not by the other applications.

Object Name: In addition to the object identifier, the OODBMS may assign one or more names to the objects that are meaningful for the programmers or the end users. The system can refer to an object by its object name. It applies certain mapping functions to determine the object identifiers and locate the desired object.

Object Lifetime: The lifetime of an object is another crucial issue to be addressed. Object lifetime determines the extant of memory or the storage time allowed to the object. Two variants of lifetime for the object are supported in the object based models. They are *transient* and *persistent*. An object, whose lifetime is transient, is allocated a memory space to be managed by the program's runtime system. When the

Space for learners:

process terminates, the memory is de-allocated. On the other extreme, an object, whose lifetime is persistent, is allocated memory space to be managed by the OODBMS runtime system. This kind of objects exists in memory after the termination of the process initiated by the application program. So, it has a long lifetime as compared to transient form.

Object Structure: The structure of an object can be either atomic or non-atomic (if the object is composed of other objects). The atomic object referred here is user-defined in nature. There is no built-in atomic object type included in OODBMS object models.

Some other important definitions useful for the demonstration of an object model are stated in the following section. The terms used here are *class*, *interface*, *struct*, *literals* and *various literal types*.

- *A class* defines both the abstract state and the abstract behaviour of the object.
- The *interface* defines only the abstract behavior of some objects.
- The *struct* defines the abstract state of some *literals*.
- *A literal* has no identifier and cannot act alone. The *literals* are embedded in objects and cannot be individually referenced. Objects and literals can be classified by their types. Although, all the elements of a given type have a common range of states and behaviors, a literal defines only the abstract state of a literal type. The value of literals does not change. Few examples of literal values are 67, 17.161576, 'P', 'Q', "GUIDOL" and "August-15, 2021". These examples are some constant numbers, characters and strings.

- In addition to the *struct* definition and the *primitive literal datatypes* (*boolean*, *char, short, long, float, double, string*), object definition languages define declarations for user-defined *collection, union and enumeration literal types*.
- Three literal types are supported by the object models. They are *atomic*, *collection* and *structured* literals.
    - **Atomic literals** correspond to the values of basic data types. Various numeric numbers, characters, Boolean values etc. are the examples of atomic literal types.
    - **Collection literals** are typically found in the ODMG object models that support literals of the following types: set<t>,

Space for learners:

bag<t>, list<t>, array<t>, dictionary<t, v> where t is a type of objects or values in the collection.

- **Structured literals** correspond to the values constructed by tuple constructor. They include the date, time, interval and timestamp as built-in structures and any other user defined structures.

1.4.1.2 Object Definition Language (ODL)

ODL is a specific kind of a language that specifies the structure of databases in object-oriented terms. ODL is an extension of Interface Description Language (IDL), which is again a component of CORBA (Common Object Request Broker Architecture). CORBA is a standard for distributed, object-oriented computing which will be discussed in the later chapters. The ODL is basically a specification language or a design language, which is used to define the specifications of object types that obey the rules of ODMG object model. This can be used like the E/R diagram used in the case of RDBMS platform. ODL is independent of any programming language and it is not used for database manipulation activities.

1.4.1.3 Object Query Language (OQL)

OQL is a query language preferred by object data management group (ODMG) for object-oriented database management purpose. OQL works closely with programming languages like C++. The embedded OQL statements within a host language return compatible objects useful for further processing. OQL's syntax is similar to SQL with additional features for object handling. This query language is designed to operate on databases described through ODL. Unlike SQL, which produces collection, OQL produces collections (sets, bags, lists) of objects. OQL fits naturally in object oriented host languages. Returned objects are assigned in the variables present in the host program and these variables are then used for further programming based manipulative works.

1.4.1.4 C++ Language Binding

Binding of ODMG implementations to C++ intends at the writing of portable C++ codes that manipulates persistent objects. This object manipulative language of C++ is abbreviated as OML. The C++ language binding includes a version of the ODL that uses C++ syntax,

Space for learners:

OQL invoking interface and some procedures for operations on OODBMS prescribed transactions.

1.4.1.5 Smalltalk Language Binding

Binding of ODMG implementations to Smalltalk focuses on the binding in terms of the mapping between ODL and Smalltalk. The Smalltalk bindings also include a mechanism to invoke OQL and required procedures for operations on databases and other transactions.

1.4.1.6 Java Language Binding

The binding between the ODMG Object Model (ODLs and OMLs) and the Java programming language is defined here. The Java language binding includes some mechanism allowing the invoking of the desired OQL and procedures for operations on ODMSs and transactions.

Space for learners:

1.4.2 Object Database and Relational Database Design

Whenever we discuss the differences between object database designs (ODB) and relational database designs (RDB), the handling of the relationships issue takes a major role.

In relational database designs, the relationships among the tuples or records are specified by the attributes with matching values. These can be termed as value references and is specified through the *foreign key* concept. Foreign keys are the values of primary key attributes in tuples of the referencing relation or table. The primary keys are limited to being atomic in nature in each record.

In object database design, the relationship issue is handled by reference attributes that include object identifiers (OIDs) of the related objects. In object database design, both single references as well as collection of references are allowed. Another notable and influencing difference between ODB and RDB design is how the inheritance is handled. These mentioned structures are built into the model, so that the mapping is achieved by using the inheritance constructs. Inheritance can be achieved through derived (:) and extends constructs. In relational design, there are several options to choose, because there is no built-in constructs for inheritance in the classic version of relational design. It is necessary to specify the operations early on in the design since they are

part of the class specifications. It is an important matter to specify the operations needed during the design phase for all types of databases. But it may be delayed in RDB design, because it is not mandatory until the implementation phase comes in force. One can easily observe one realistic difference between the relational model and the object model of data in terms of behavioral specifications. Although relational data models do not compel or encourage the database designers to set some valid behaviors or operations, this is an implicit requirement in the case of object models.

Space for learners:

1.4.2.1 Mapping of an Enhanced Entity Relationship (EER) Schema into an Object Database (ODB) Schema

The correlation of EER schemas and ODB schemas is simple, because the ODB schemas provide support for inheritance. Once the mapping has been completed, the operations need to be added to ODB schemas. It is because the EER schemas do not include any operations like ODB. The mapping of EER into ODB schemas can be exhibited using the following steps.

Step -1

- Creation of an ODL class for each EER type.
- Multi-valued attributes are declared by sets, bags or lists.
- Composite attributes are mapped into tuple constructors.

Step – 2

- Add reference attributes for each binary relationship into the ODL classes that participate in the relationship.
- Relationship cardinality is set as single-valued for 1:1 and N:1 types and set- valued for 1:N and M:N types.
- Relationship attributes are created through the use of tuple constructors.

Step - 3

- Include the operations corresponding to each class.

- EER schema does not provide these operations and it must be added to the database design by choosing it from the original requirements.
- The associating constructor and destructor operations must also be included.

Step - 4

- Inheritance relationships can be specified via *extends* clause.
- An ODL class that corresponds to a sub-class in the EER schemas inherits the types and methods of its baser-class in the ODL schemas.
- Its non-inherited attributes, relationship references and operations are specified as mentioned in the earlier steps.

Step - 5

- Weak entities can also be mapped in the same way as the regular entity types.
- Non-participating weak entities in any relationships may alternatively be presented as composite multi-valued attribute of the owner entity.
- The attributes of the weak entity are included in the struct <... > construct.

Step - 6

- Map categories (union types) to object definition language.
- May follow the same mapping used for EER-to-relational mapping.
- Declare a class to represent the category.
- Define the 1:1 relationship between the category and each of its base-classes.

Step – 7

- Map multi-dimensional cardinality relationships whose degree is greater than 2.

Space for learners:

- Each relationship is mapped into a separate class with appropriate reference to each participating class.

Space for learners:

1.5 OBJECT RELATIONAL DATABASE SYSTEMS

Object–relational database systems are commonly termed as Object–relational database management systems (OORDBMS). OORDBMS is an object-oriented version of the traditional relational database management systems (RDBMS). This is a kind of a hybrid approach capable of handling the object oriented as well as relational aspects of DBMS, which well fits with the current industry requirements.

1.5.1 Relational Database Management System (RDBMS)

RDBMS is a simply the relational version of traditional DBMS, which incorporates the terms relations, tables, attribute, columns, integrity, security etc. into its operational procedures. RDBMS deals with a number of tables together to store, edit, update and delete data considering the normalization aspects like 1NF, 2NF, 3NF and BCNF forms. Standard SQL statements are used to operate on RDBMS. Various commonly used RDBMSs are Oracle, Microsoft's SQL server, MySQL etc. Although, most of the needs of a common database user are addressed by these softwares, the object-oriented aspects could not be incorporated here. This is the reason why the object-oriented relational database management system is becoming the need of the hour. The later section is going to elaborate these aspects followed up by sating the differences between OODBMS and RDBMS.

1.5.2 History of Object Relational Database System

The Object–relational database system or OORDBMS came into light in the early 1990s. This trend comes into existence by extending the relational database concepts with the addition of the concept of *object*. The industry experts aimed to get hold on a declarative query-language based upon predicate calculus as a vital component of OORDBMS. Two most notable research projects viz., Illustra and PostgreSQL was

brought into reality by Postgres (UC Berkeley) during this time. In the mid of 1990s, early commercially available products were released. These releases include various products like Illustra (IBM), *Omniscience* (Oracle) and UniSQL (KCOMS). The Ukrainian developer Ruslan Zasukhin, who is the founder of Paradigma Software, Inc. developed and released the first version of Valentina database in the mid of 1990s, which was used as C++ SDK. After less than a decade of time, PostgreSQL had become a commercially available database and has become the basis for several currently available products incorporating OORDBMS features. The experts in the domain started referring these products as *object oriented relational database management systems* or OORDBMS. Many of the ideas of early object relational database efforts have largely been incorporated into SQL: 1999 via specific structured types. For example, IBM's DB2, Oracle database, and Microsoft's SQL Server are claiming to support most OORDBMS requirements and do so with a varying degree of success.

SQL statements are written in RDBMS like this-

```
CREATE TABLE Customers (
    Id   CHAR(10)  NOT NULL PRIMARY KEY,
    Surname  VARCHAR(30) NOT NULL,
    FirstName  VARCHAR(30) NOT NULL,
    DOB  DATE   NOT NULL                    [# DOB : Date of Birth]
);

SELECT InitCap(Surname) || ', ' || InitCap(FirstName)
FROM Customers
WHERE Month(DOB) = Month(getdate())
AND Day(DOB) = Day(getdate());
```

Standard SQL databases allow customized functions also, which allow the following type of query-

```
SELECT Formal (Id)
FROM Customers
```

Space for learners:

```
WHERE Birthday (DOB) = Today();
```

In OORDBMS, queries containing user-defined data-types and expressions like BirthDay() are seen as mentioned below-

```
CREATE TABLE Customers(
    Id  Cust_Id  NOT NULL  PRIMARY KEY,
    Name  PersonName  NOT NULL,
    DOB  DATE  NOT NULL
);
SELECT Formal( C.Id )
FROM Customers C
WHERE BirthDay ( C.DOB ) = TODAY;
```

The object relational models can offer another interesting capability. Here, the database can make use of the relationships between the data to easily fetch the related records. For example, in an address book software application, an additional table is added to the existing ones to hold the addresses of customers. Using a traditional RDBMS, collecting information for both the user and their address requires a "join" as mentioned below-

```
SELECT InitCap(C.Surname) || ', ' || InitCap(C.FirstName), A.city
FROM Customers C join Addresses A ON A.Cust_Id=C.Id
WHERE A.city = "New York";
```

The above query when applied in an object–relational database appears in a simpler way as mentioned below-

```
SELECT Formal ( C.Name )
FROM Customers C
WHERE C.address.city = "New York";
```

Space for learners:

1.5.3 Object-Oriented Relational Database Management System (OORDBMS)

Space for learners:

An object-relational database is maintained by a relational database management system with an associating object-oriented database model, where all data and data models are created treating them as objects. Data abstraction, data hiding, early binding, late binding, polymorphism and inheritance like properties are directly supported in the database schemas and the associating query languages support the object based data access. *Oracle* is one of the popular RDBMSs, which meets the industry standards. The object-relational database systems are an attempt to merge the two dissimilar trends together. It can be visualized as an object database expansion of a relational model resulting in a hybrid design. One of the most visible aspects that we might observe is in the addition of object database features in the SQL revision. But, the tough part of a relational model immerges when someone tries to describe complex objects.

The object-oriented relational database mechanism gains its importance with the introduction of the type constructors describing row types, array type being replaced by collections, sets and lists. The creation of derived mechanisms for specifying object identity, encapsulation and inheritance is also helping OORDBMS to gain its importance. It is to be noted that the core technology used in OORDBMS is based on relational models. The commercial products (e.g., Microsoft SQL Server) have simply added a layer of some object-oriented principles on top of the relational database management system. The translation of object-oriented mechanism into relational mechanism is one of the challenging tasks for typical OORDBMS. This problem is typically addressed by an object-oriented application that does the communication between the object-oriented applications with the underlying relational databases.

Both relational and object-oriented mechanisms are having a lot of differences in terms of their underlying principles. This is the reason why this model tries to negotiate among these two techniques to adopt some intermediate measures for the sake of developer's convenience. One of the very important reasons is to permit the storage and retrieval of objects in a way how RDBMS functions. This act provides an

extensive liberty to query languages to work on the object-oriented principle. Some of the common implementations in this regard are the Oracle Database, PostgreSQL, and Microsoft's SQL Server. IBM DB2 also supports objects and can be considered as OORDBMS.

In OORDBMS, the approach is essentially that of relational databases, where the data resides in the database and is manipulated collectively with queries through a query language. But, in OODBMS, where the database is essentially a persistent object store for software written in an object-oriented programming language, a programming API is solely responsible for storing and retrieving of objects. In this case, a very little or no specific support presents for query languages.

The basic need of object–relational database arises from the fact that both relational and object databases have their individual advantages and drawbacks. Although, the object-oriented databases allow sets, lists, arbitrary user-defined data types and nested objects, they do not provide any mathematical base for in-depth analysis. The basic goal for the object–relational database is to bridge the gap between relational databases and the object-oriented modeling techniques. The commonly used programming languages such as C++, C#, Java and Visual Basic.NET are seen implementing these extensive features of object-relational databases. Further, the object–relational DBMS or OORDBMS allows software developers to integrate user-defined data types and methods that apply to them into the DBMS. Some of the leading features or characteristics of OORDBMS are *Complex data, Type inheritance* and *Object behavior.*

Complex data creation is based on basic schema definition through the user-defined types. Structured complex data are when stored in a hierarchy; it offers an additional property termed as *type inheritance*. That is, a structured type can have subtypes that reuse all of its attributes and contain additional attributes specific to the subtype. Finally, the *object behavior* is related with the access to the program objects. Such program objects must be storable and transportable for database processing. This is the reason why they are usually named as persistent objects. Inside a database, all the relations with a persistent program object are relations with its object identifiers. The mentioned points above can be addressed in a proper relational system, although the SQL standard and its implementations enforce arbitrary restrictions

Space for learners:

and some amount of additional intricacy. Extension of the data model with custom data types and methods is possible in a properly arranged relational system.

Space for learners:

1.5.4 Comparative Analysis of RDBMS and OORDBMS

The comparative analysis of a typical RDBMS with the OORDBMS helps understanding the changes made in the OORDBMS.

Table 1.1: Comparative Analysis of RDBMS and OORDBMS

RDBMS	OORDBMS
Ensures only the data independence part	Ensures data independence as well as data encapsulation and abstraction of data
Data can only be recognized without affecting the mode of using it	Data as well as class/object can be recognized without affecting the mode of using it
Stores only the data	Stores not only the data but also the methods imposed on that data
Data can be partitioned depending upon the user's requirements and specific user's applications	The data can be used in direct access mode and also through the class/object methods and sometimes the entire data can be made public using specific access controls
Users can perceive data as columns, rows or tuples (records) and tables	Apart from handling complex structure of data this system can handle relational data

Thus, after all these discussions, an OORDBMS can be understood as a DBMS, but with the extended relational and object-oriented capabilities. It is because of the functional differences among these two extending approaches, they are to proceed in a hand-in-hand strategic and somewhat compromising approach.

CHECK YOUR PROGRESS

Multiple choice questions:

1. Object databases are based on
   i) Relational approach
   ii) Object based approach
   iii) Both *(i)* and *(ii)*
   iv) None of these

2. The term attribute refers to a
   i) Record
   ii) Row
   iii) Column
   iv) Key

3. Which of the following can be defined using ODL?
   i) Structure
   ii) Attribute
   iii) Operation
   iv) All of above

4. Which of the following belongs to an atomic literal?
   i) String
   ii) Boolean
   iii) Long
   iv) All of above

5. Which among the following is/are not Object Based Database(s)?
   i) Cache
   ii) Foxpro
   iii) Wakanda
   iv) Both (i) and (iii)

State whether *True* or *False:*

6. A single programming paradigm acts behind a single programming language.

7. A class is an instance of an object in OOP.

8. ODMG looks after the object models in an OODBMS.

9. MS SQL Server does not support OOP principles in any of its versions.

10. OORDBMS works in the principles of OOPs as well as the relational models.

Space for learners:

1.6 SUMMING UP

Space for learners:

- DBMS is a software platform, where data are stored, managed and retrieved as per user's requirement through some standard language queries like SQL.
- RDBMS is a similar system like DBMS which also store, manage and retrieve data when needed by the users, but is has the additional capability of maintaining relational tables or data.
- ODMG stands for Object Data Management Group. It is a consortium responsible for the monitoring of object oriented database management activities.
- OODBMS is a DBMS system, where the data are stored, managed and retrieved considering most of the data as objects is called the OODBMS. Object oriented features like inheritance, polymorphism are applicable here.
- OORDBMS is a RDBMS system with the object oriented extension which is capable of implementing object oriented features like class and object, inheritance, polymorphism etc. in addition to the classic RDBMS features.
- ODL is a specific kind of a language that specifies the structure of databases in object-oriented terms.
- OQL is a query language preferred by object data management group (ODMG) for object-oriented database management purpose.

1.7 ANSWERS TO CHECK YOUR PROGRESS

Multiple choice questions:

1. (ii) 2. (iii) 3. (iv) 4. (iv) 5. (ii)

State whether *True* or *False:*

6. False 7. False 8. True 9. False 10. True

1.8 POSSIBLE QUESTIONS

Space for learners:

Short answer type questions:

1) What is the difference between an object and a literal in the object oriented data model (OODM)?
2) What is an object? What is an object model with reference to ODMG standards?
3) What are the main difference between designing a relational database and an object database?
4) Differentiate between:
   i) Interface and Class
   ii) Atomic object and Collection object
   iii) Object identifier and Object lifetime
   iv) Persistent object and Transient object
5) What is the significance of ODL in OODBMS?

Long answer type questions:

1) Explain the major specifications mentioned in ODMG 3.0 standard.
2) Describe the differences and similarities between objects and literals in the ODMG object model?
3) Describe the steps involved in mapping the EER schema into ODB schema.
4) Explain in detail the OORDBMS concept with the introduction to all its organizing components.
5) Describe in detail the differences between RDBMS and OORDBMS.

1.9 REFERENCES AND SUGGESTED READINGS

- M. Stonebraker, "Inclusion of New Types in Relational Database Systems", In Proc. of the International Conf. on Data Engineering (1986), pages 262–269.
- M. Stonebraker and L. Rowe, "The Design of POSTGRES", In Proc. of the ACM SIGMOD Conf. on Management of Data (1986), pages 340–355.
- M. Atkinson, et.al., "The Object-Oriented Database System Manifesto",

In Proceedings of the "First International Conference on Deductive and Object-Oriented Databases", pages 223-40, Kyoto, Japan, December 1989.

- W.Kim And Lochovsky (Eds), Object-Oriented Concepts, Databases, and Applications, Addison-Wesley (Reading MA), 1989
- https://en.wikipedia.org/wiki/Comparison_of_object_database_management systems
- https://en.wikipedia.org/wiki/Object_database
- http://www.odmg.org

Space for learners:

UNIT 2: DISTRIBUTED DATABASE

Space for learners:

Unit Structure:

2.1 INTRODUCTION

The database is a collection of structured information. Among other database systems, a distributed database is one where files are stored in different sites or systems. This unit will give an overview of the distributed database Management System (DDBMS). The unit shows the uses of distributed databases. Data fragmentation, replication, and allocation are very much important in a database. These are also explained in detail in this unit by considering the example of distributed database. Types of the distributed database are also explained in this unit by considering the examples. Query processing and data recovery of the distributed database are shown by taking the database example.

2.2 UNIT OBJECTIVES

After going through this unit, you will be able to know

i) About the distributed database

ii) About the types of the distributed database
iii) About the data fragmentation, replication, and allocation in a distributed system.
iv) About the query processing database distributed system.
v) About the concurrency and recovery in a distributed database.

Space for learners:

2.3 DISTRIBUTED DATABASE

The database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network. The application service layer of the distributed system provides services to the user. The users are unknown about the distributed storage structure of the system. They think that one single database is present in the system to provide services to the user. Data is distributed among the system through the communication network and it is controlled by the Distributed Database Management System (DDBMS).

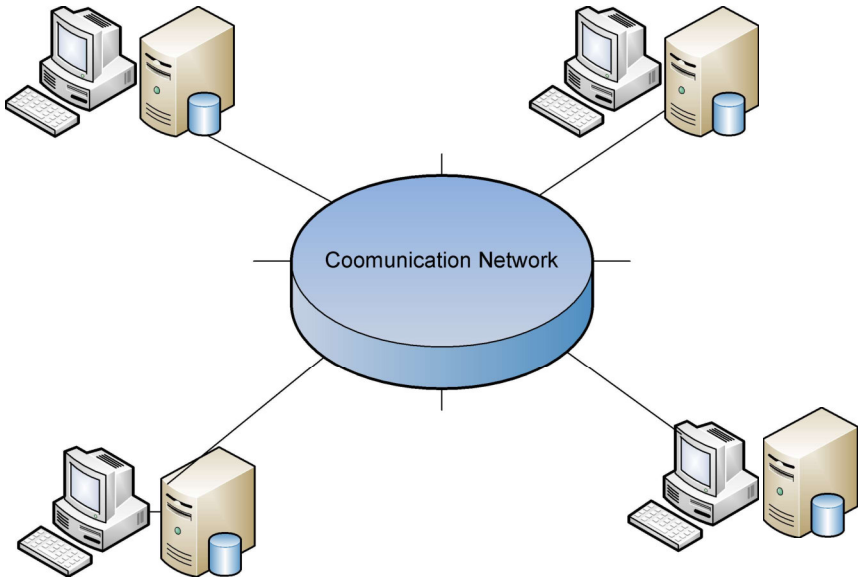


Fig. 2.1. Architecture of DDBMS

In Fig. 2.1, four different systems are interconnected through the communication network. This type of system is known as a distributed system which is also known as a loosely coupled

system. In this type of system, the data is distributed among the system. That is the reason the database of this type of system is known as a distributed database. Every DDBMS has some features.

I. Databases of the DDBMS are interlinked logically and they are connected through a communication network. Often, the DDBMS act as a single database for the user.

II. Data is physically stored in multiple sites and the data is managed by a local DBMS in the site which is independent of the other sites.

III. A distributed database is not a loosely connected system.

IV. A distributed database integrates transaction processing.

V. DDBMS synchronizes the distributed database periodically for which it is transparent to the users.

Every distributed database has to build with some goals and these are as follows.

i) Reliability: In DDBMS, if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.

ii) Availability: In DDBMS, sites or systems are available to provide reliability to the system. If one distributed system fails, other sites can give service, and it maintains the availability of the systems.

iii) Performance: Performance of the DDBMS can be achieved by distributing data or information over different sites which are located in different locations. So, the databases are available to every location which is maintained through the communication channel.

CHECK YOUR PROGRESS-I

1. What do you mean by distributed database?
2. What is reliability in DDB?
3. What are the goals of a distributed system?

2.4 DATA FRAGMENTATION IN DDB

Fragmentation is a normal process of diving the database into different tables in DBMS. In a distributed database, the entire

Space for learners:

database is divided into different subtables or sub relations so that each subtable or sub relation can be saved in different sites of the distributed system. These subtables or sub relations are the logical units of the DDBMS. The fragmentation is done in such a way that the subunits give the actual distributed database after combining it. Let's, you have a table T in your distributed system and it is fragmented into different sub tables t1, t2, t3, ----, tn. These fragments should have sufficient information, so that it will restore the original table after combining the t1, t2, t3, ---, tn using the UNION or JOIN operation. These subtables are known as the fragments and the process is known as data fragmentation in DDBMS. The fragments are independent of each other's and user are concerned about the data fragmentation. This is known as fragmentation transparency.

The distributed data fragmentation process has some advantages:

I. As the data is fragmented and can be stored locally, the performance of the DDBMS will increase.

II. Due to the local data store in the local sites, local query optimization is possible in DDBMS.

III. Fragmentation helps to main the security and privacy of the local system which will help to main the overall security of the DDBMS.

You have 3 methods for data fragmenting of a table and they are.

i) Horizontal Fragmentation.

ii) Vertical Fragmentation.

iii) Hybrid Fragmentation.

Space for learners:

2.4.1 Horizontal Fragmentation

Horizontal fragmentation allows dividing a table horizontally into subsets of tables. It means that it will divide the table row-wise(tuple). Let's you have a table IDOL as follows.

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2019001	D	BSc. IT
2020002	B	MSc. IT

2020003	C	MSc. IT
2019002	E	BSc. IT

Now, you can divide this table into different fragments based on different conditions such as branches. For example,

i) you may Fragment 1 where Branch is BSc. IT.
ii) you may Fragment 2 where Branch is MSc. IT.

Now, the fragment outputs are presented below.

Fragment 1 =

S_Roll_No	S_Name	Branch
2019001	D	BSc. IT
2019002	E	BSc. IT

Fragment 2 =

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT

Now if you combine these two fragments (fragment 1 and fragment 2), you will get the original table after performing the union operation between fragment 1 and fragment 2 as follows.

IDOL = fragment 1 U fragment 2.

In DDBMS, the fragments are saved in different sites as follows. In Fig. 2.2, fragment 1 is saved in site A where fragment 2 is saved in site B.

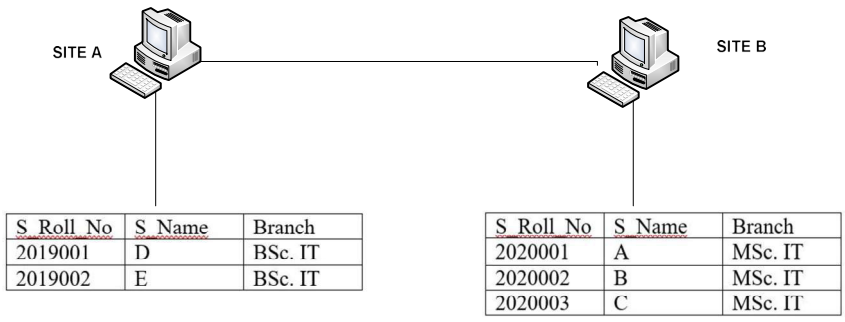


Fig. 2.2: Horizontal Fragmentation in DDBMS.

Space for learners:

2.4.2 Vertical Fragmentation

Vertical fragmentation divides the table column-wise (attribute). It is more complex than horizontal fragmentation. For the reconstruction of the original table from the fragment, the primary key should be available in all the fragments. The reconstruction is doing using join. For the above table IDOL database, you can create the follows vertical fragmentation.

Vertical Fragmentation 1 =

S_Roll_No	S_Name
2020001	A
2019001	D
2020002	B
2020003	C
2019002	E

Vertical Fragmentation 2 =

S_Roll_No	Branch
2020001	MSc. IT
2019001	BSc. IT
2020002	MSc. IT
2020003	MSc. IT
2019002	BSc. IT

In vertical fragmentation 1 and vertical fragmentation 2, one filed is common i.e the primary key of the IDOL table. It is required to perform the join operation between the fragments. You can join the two fragments to get back the original table IDOL as follows.

Π_{IDOL} (T1 ⋈T2).

In DDBMS, the vertical fragments are saved in different sites as follows. In Fig. 2.3, fragment 1 is saved in site A where fragment 2 is saved in site B.

Space for learners:

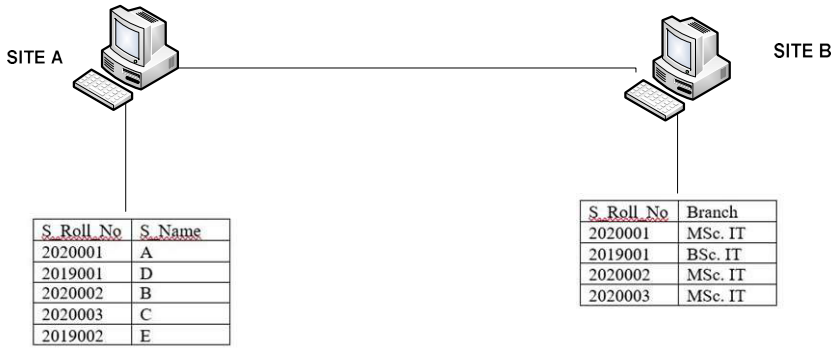


Fig. 2.2: Vertical Fragmentation in DDBMS.

2.4.3 Hybrid Fragmentation

It is a combination of horizontal and vertical fragmentation. This fragmentation can be done in two ways.

I. The first method is to first create horizontal fragments and then create vertical fragments.

II. The second method is to first create vertical fragments and then create horizontal fragments

For the above IDOL table, the hybrid fragmentation can be found as follows.

S_Name	Branch
A	MSc. IT
B	MSc. IT
C	MSc. IT

In this fragmentation, the first horizontal fragmentation followed by the vertical fragmentation has been done.

Space for learners:

Space for learners:

CHECK YOUR PROGRESS-II

4. True or False
   i) Vertical fragmentation divides the table column-wise (attribute).
   ii) Horizontal fragmentation allows dividing a table horizontally into subsets of tables
5. Let's you have a table COURSE as follows.

S_Roll_No	S_Name	Course
2020001	A	MSc. IT
2019001	D	BSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT
2019002	E	BSc. IT

   i) Create one horizontal fragmentation based on MSc. IT Course
   ii) Create one vertical fragmentation based on Roll No and Course = Bsc. IT

2.5 DATA REPLICATION AND ALLOCATION

The process of storing data or information in more than one site or system in a distributed system is known as data replication. It is useful in improving the availability of data. Data replication copying the data from the database of one system to another system. Due to this process, users can send the same data without any inconsistency. The main goal of data replication is to increase the availability of the data and also to increase the query processing technique. Two types of data replication are present.

i) **Synchronous Data Replication**: In this type of replication, once the changes are made in a table of the database, the data replication is done immediately.
ii) **Asynchronous Data Replication**: In asynchronous replication, the data replication is done after the commit operation of the database.

Apart from the above data replication, there are another few data replications in a distributed database.

i) **Transactional Replication**: Transactional replication is generally used in server-to-server communication. In this replication, a full copy of the database is present with one system and that system gets the update notification from the other system once the data changes. Data replication is done in real-time, so it gives a consistency guarantee. For example, Azure SQL.

ii) **Snapshot Replication**: In this replication, a snapshot of the database is sent to one database from another database. Data is not updated continuously. Data is updated infrequently at a specific time. It is more complex than transactional replication. For example, SQL Server replication.

iii) **Merge Replication**: In this replication, data of one database is combined with another database. In this type of replication, the data is updated from both databases, so hard to main consistency and concurrency. For example, Server and Client Communication (SQL server).

Data replication in DDBMS happens in different modes. They are as follows.

i) **Full Replication**: In this mode, a full copy of the database is present at every site of the distributed system. This mode increases the availability of the data in the system, and the user gets the highest experience from it. It is hard to main the concurrency.

ii) **No Replication:** Here, the data is divided into different fragments and each fragment is present at only one site which is located in different locations. Data availability is less than the full replication but concurrency can be controlled.

iii) **Partial Replication:** Here some of the data fragments of the database are replicated but some are not. Data replication is depending on the demands of the respective data fragments.

Data allocation is a process to decide where exactly you want to store the data. It involves at per which data has to be stored at what location. The data allocation technique allocates data fragments to a site in a distributed database. Each data fragment or

Space for learners:

its replication can be stored in the particular site of a system. The process of storing data in a site is known as data allocation. The sites and numbers of data replication depend on the demand of the data fragments. The choice of sites and the degree of replication depend on the system performance and availability and also depend on the number of transactions submitted on the site. If the user demands high availability of data, then full replication is a good choice for this allocation. Otherwise, if a fragment of data is required then partial replication can be used to allocate the data.

Three main data allocation methods are there and they are as follows.

i) Centralized: Here entire database is stored in a single site. No such data distribution or replication occurs in this process.
ii) Partitioned: In this technique, data is divided into different fragments and those fragments are stored in different sites of the distributed systems.
iii) Replicated: In this technique, a copy of the database is present in a different location and it is accessed from those locations.

Space for learners:

2.6 TYPES OF DDB SYSTEM

There are two types of distributed databases are found and they are homogenous database and heterogeneous database.

i) Homogeneous Database

In a homogeneous database, the physical and logical structures of the database are identical for all the systems. It means that OS, DBMS, and software are the same for that system. Hence, it is easy to manage the homogenous distributed database. For example, Oracle Database server.

ii) Heterogeneous Database

In a heterogeneous distributed database, the physical and logical structures of the database are not the same. The sites use different schema and software. It is hard to manage concurrency and transactions in a distributed system. Here, one site of the distributed system may be completely unaware of the other sites of the systems. For example, Oracle8.

2.7 QUERY PROCESSING IN DDB

Space for learners:

In a distributed database system, query processing is done at the end of the user site and server site. A query comes from the user site, so it is checked and optimized at the user site i.e. it is at the local level. The query comes to the server, so it is processed and optimized at the server site i.e., it is at the global level.

In a homogeneous distributed database, when a query comes from a user site, it will be able to manage the query easily as the sites have the same physical and logical structure. But in heterogeneous systems, it will not able to manage the work easily. So, there should be some techniques to handle queries in heterogeneous system databases. There are two types of mechanisms in the heterogeneous system to manage such situations.

i) Multi-Database

   In this method, a dynamic schema is created for the respective databases. If a user site uses a database, then a dynamic schema is created to connect the database D. Due to this schema, the user query is flexible with the database.

ii) Federated mechanism

   In this method, a global schema is used to access the database. It means that a centralized schema is used to access all the databases of the distributed system. This global schema will work properly even though the data is fragmented and distributed over different sites.

When a federated mechanism is used, a few of the things have to manage during the database access. They are presented below.

i) Data Models

   During the time global schema, the schema should take care of the data model. Because distributed database means different databases with their physical and logical structure. So, the federated schema should be compatible with all these types of systems and also should handle the query.

ii) Constraints

   Each database of the distributed system has its process of defining the data constraints and has its method of accessing

the data. So, the federal schema should handle these constraints.

iii) Query Language

In a distributed database, the databases are varying from site to site. So, the query languages are also varied from site to site. Hence federated schema should develop a common language that is compatible with all the query languages.

iv) Data Transfer Cost:

In a distributed database, databases are distributed. So, the table of the databases is also distributed. Even some tables are fragmented. So, during the time of query processing; it may need to access the tables at the different databases or different locations. This demands a request and transfer cost for the data which needs to optimize.

To explain data transfer cost, let's you have two distributed database tables namely IDOL_EMP and IDOL_DEPT. The IDOL_EMP has a table ÉMP which is present in one location (location 1) of the distributed system, and IDOL_DEPT has another table DEPT in another location (location 2) of the distributed system. The EMP contains the basic information of the employee where the DEPT table contains the name of the department where the employee works. Let's you have 500 data of size 50 bytes in your EMP table where DEPT table has 10 data of size 10 bytes. Consider you have processed a query to find the name of the employee and department from another location (location 3). The result of this query will include 500 records, assuming that every employee is related to a department. Suppose that each record in the query result is 40 bytes long. In this situation, you can execute your query based on the three costs, and accordingly, you can choose the optimized cost.

CASE I. You are executing your query from location 3. For this case, the cost is as bellow.

i) Cost of transferring EMP data: 500 records * 50 bytes = 25,000 bytes.

ii) Cost of transferring DEPT data: 10 records * 10 bytes = 100 bytes.

Space for learners:

iii) Therefore, total cost = 25,000 bytes + 100 bytes
= 25,100 bytes

CASE II: You can shift the data of the EMP table from location 1 to location 2 and then you process it and transfer the data to location 3. For this case, the cost is as bellow.

i) Cost of transferring EMP data: 500 records * 50 bytes = 25,000 bytes

ii) Cost of transferring the result: 500 records * 40 bytes = 20,000 bytes.

iii) Therefore, total cost = 25,000 bytes + 20,000 bytes
= 45,000 bytes

CASE III: You can shift the DEPT data of the EMP table from location 2 to location 1 and then you process it and transfer the data to location 3. For this case, the cost is as bellow.

i) Cost of transferring DEPT data: 10 records * 10 bytes = 100 bytes

ii) Cost of transferring the result: 500 records * 40 bytes = 20,000 bytes.

Therefore, total cost = 100 bytes + 20,000 bytes = **20,100 bytes**

Now, if you compare the cost of CASE I, CASE II, and CASE II, the cost of CASE III is the minimal one and it is optimized. Using this method, you can perform your query in the distributed database at a minimal cost.

2.8 CONCURRENCY AND RECOVERY IN DDB

During the time of concurrency control and recovery distributed databases face lots of issues. They are presented below.

i) Multiple copies of data:

A distributed system may have a copy of the database in each site to increase the availability of the system. To make a copy consistent and to maintain consistency among the copies of the database, concurrency and recovery are important, but it is not easy to maintain consistency.

ii) Failure of a site:

Space for learners:

In a distributed system, the database of one site may fail. But the DDBMS should work with the other sites and it will try to recover the sites and make its date up to date.

iii) Failure of Communication Network:

The DDBMS must deal with the communication failure and will try to maintain the concurrency and recover the sites as soon as possible. If network portioning occurs due to network failure, then it is hard to recover the sites and maintain consistency.

iv) Distributed Commit:

The problems occur when a commit transaction is done in DDBMS where the database is present in a failed system. The two-phase commit protocol is often used to deal with this problem.

v) Distributed Deadlock:

Sometimes deadlock may occur in a distributed system. So, it is necessary to main consistency and recovery in the deadlock system.

The techniques which deal with concurrency control in DDBMS are explained below.

I. Lock based protocol:

When two transactions are present in the database, a read-write lock can apply in one transaction to avoid the concurrency issue where others can access the data. This lock

II. Shared lock system (Read lock):

The shared lock system is a read lock. The lock is shared between the transaction. Any one of the transactions can activate the shared lock for reading purposes.

III. Exclusive lock:

In this technique, an exclusive lock is activated for a transaction for the read and write operation. In this technique, no other lock can apply for the read and write operation on the same data.

Space for learners:

Lock-based concurrency protocol locks the data. A lock is a variable that controls the read-write operation on data. It is two types.

i) One phase Locking Protocol:

   In this technique, a lock is applied by a transaction on data before it uses and releases after the transaction is complete.

ii) Two-phase locking protocol:

   In the two-phase locking protocol, a transaction adopts all the locks in the first phase and does not release any locks until finish all read and write operations. In the second phase, the transaction releases all the locks and never requests any locks.

Recovery is the most important process in a DDBMS. It is required to recover the information from a site. The recovery is required due to the following reasons.

i) The receiver site may down
ii) The location of the receiver site may crash.
iii) The communication link between the sender and receiver site may break.

A two-phase commit protocol is used to overcome the issue of the data recovery on DDBMS. This atomic protocol coordinates the process of DDBMS which decides to commit or terminate a transaction. It provides the automatic recovery option in case of a site failure. The original place of transaction is known as coordinator and other places of the transaction are known as a cohort. The protocol executes in two phases.

i) Commit request: In the commit phase, the coordinator prepares the list of cohorts and asks to commit the transaction.

ii) Commit phase: Based on the responses from the cohorts, the coordinator can decide to commit or terminate a transaction.

Space for learners:

CHECK YOUR PROGRESS-III

6. All sites in a distributed database commit at exactly the same instant. TRUE/FALSE
7. Fill in the blanks.
   i) The real use of the Two-phase commit protocol is _____.
   ii) Read one, write all available protocol is used to increase _____ in a distributed database system.
   iii) Commit and rollback in DDB are related to .....
   iv) If a distributed transactions are well-formed and 2-phasedlocked, then ..... is the correct locking mechanism in distributed transaction as well as in centralized database.
   v) A distributed transaction can be ..... if queries are issued at one or more nodes.

Space for learners:

2.9. SUMMING UP

- The distributed database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network.
- Data in DDB is physically stored in multiple sites and the data is managed by a local DBMS in the site which is independent of the other sites.
- A distributed database integrates transaction processing.
- In DDBMS, if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.
- Fragmentation is a normal process of diving the database into different tables in DBMS. In a distributed database, the entire database is divided into different subtables or sub relations so that each subtable or sub relation can be saved in different sites of the distributed system.

- You have 3 methods for data fragmenting of a table and they are.
  - Horizontal Fragmentation.
  - Vertical Fragmentation.
  - Hybrid Fragmentation
- The process of storing data or information in more than one site or system in a distributed system is known as data replication. It is useful in improving the availability of data.
- Two types of data replication are present.
  - **Synchronous Data Replication**: In this type of replication, once the changes are made in a table of the database, the data replication is done immediately.
  - **Asynchronous Data Replication**: In asynchronous replication, the data replication is done after the commit operation of the database.
- Data allocation is a process to decide where exactly you want to store the data. It involves at per which data has to be stored at what location.
- There are two types of distributed databases are found and they are homogenous database and heterogeneous database. a) **Homogeneous Database b) Heterogeneous Database**
- In a distributed database system, query processing is done at the end of the user site and server site. A query comes from the user site, so it is checked and optimized at the user site i.e. it is at the local level. The query comes to the server, so it is processed and optimized at the server site i.e. it is at the global level.
- During the time of concurrency control and recovery distributed databases face lots of issues. They are presented below.
  - Multiple copies of data,
  - Failure of a site,
  - Failure of Communication Network,
  - Distributed Commit,
  - Distributed Deadlock.

Space for learners:

- A lock is a variable that controls the read-write operation on data. It is two types.
  - One phase Locking Protocol: In this technique, a lock is applied by a transaction on data before it uses and releases after the transaction is complete.
  - Two-phase locking protocol: In the two-phase locking protocol, a transaction adopts all the locks in the first phase and does not release any locks until finish all read and write operations. In the second phase, the transaction releases all the locks and never requests any locks.

Space for learners:

2.10 ANSWERS TO CHECK YOUR PROGRESS

1) The distributed database is a collection of structured information. Among all database systems, a distributed database is one where files are stored in different computer systems or sites. These sites are connected through a communication network.
2) In DDBMS, reliability means if one of the systems fails, then other systems will provide the service to the user. The other system can complete the task of the failure system.
3) The goals of DDB are as follows.
   I. Reliability
   II. Availability
   III. Performance
4) i) TRUE ii) TRUE
5) i)

S_Roll_No	S_Name	Branch
2020001	A	MSc. IT
2020002	B	MSc. IT
2020003	C	MSc. IT

   ii)

S_Roll_No	Branch
2019001	BSc. IT
2019002	BSc. IT

6) FALSE

Space for learners:

7)

i) Atomicity, i.e, all-or-nothing commits at all sites

ii) Both Availability and Robustness

iii) Data Consistency

iv) A two-phase locking.

v) partially read-only

2.11 POSSIBLE QUESTIONS

Short answer type questions:

1. What is a distributed system?
2. What is distributed database?
3. What are the goals of the distributed database?
4. What is the reliability and availability of distributed database?
5. Difference between one phase and two-phase locking protocol.
6. What are the types of distributed database systems available?
7. What are the different modes of data replication in a distributed system?
8. Difference between lock-based and shared lock systems.
9. What is data replication in DDBMS? What are the types?

Long answer type questions:

1. Explain the distributed database system with an example.
2. Explain with examples the fragmentation of tables in the distributed system.
3. How are concurrency and recovery achieved in the distributed database?
4. Explain data replication and allocation in DDBMS.
5. Explain the query processing in DDBMS.

2.12 REFERENCES AND SUGGESTED READINGS

i) Principles of Distributed Database Systems. Author: M. Tamer Özsu.

ii) Distributed System: Concepts, Design, and Applications Publisher: O, Reilly, Author: S.K.Singh

Space for learners:

UNIT 3: IMAGE AND MULTIMEDIA DATABASE

Space for learners:

Unit Structure:

3.1 INTRODUCTION

This unit gives an overview of the multimedia database, especially about the image database. Image means the collection of pixels. Pixels have information about the images. The process of storing images in a database is discussed in this unit. The unit also discusses the contents of the multimedia database. Challenges of the multimedia database are also discussed in this unit. The contents of the multimedia database are also pointed in this chapter. Finally, the different applications of the multimedia database are reported in the unit.

3.2 UNIT OBJECTIVES

After learning this unit, you will be able to learn

i) About the definition of multimedia database

ii) About types of multimedia database including an image.

iii) About image and multimedia database.

iv) About the challenges and contents of the multimedia database.

v) About the challenges of the multimedia database.

vi) About the applications of the multimedia database.

Space for learners:

3.3 CONCEPT OF IMAGE

An image is multimedia data. It consists of the pixel. The pixel of an image contains all the necessary information about the image. An image may be color, grayscale, or black and white. You can extract the information of color from the pixel of an image. Apart from color, other features such as texture and shape are also possible to extract from an image. These features can be stored in a database. Images are used in different fields, so it is necessary to store the images in the database. The database where images are stored is known as a multimedia database.

3.4 IMAGE AND MULTIMEDIA DATABASE

An image can not directly store in a database using a standard SQL insert command. The embedded SQL is used to insert the images into a database. A database should support an image to insert an image in the database. The images are stored in binary form in the cell of a table of a database and the data type of the cell is Binary Large Object (BLOB). It is a MySQL data type that is not only used to store the image data but also used to store the other data type. For tightly coupled database such as employee database, student database needs to upload the image in the database, so this type of databases are known as multimedia database and storing of an image one of the part of this database.

Let's explain the BLOB in MySQL using python. Her, you will learn about the process of insertion and deletion of multimedia files such as images, video, or songs in a multimedia MySQL database using python. To Store and retrieve multimedia data, i.e BLOB data in a MySQL table, you should have a table containing binary data or you can update your table by inserting one extra column in the

database for the BLOB data. You can execute the following queries for the BLOB data.

i) **Table Creation Query:** CREATE TABLE `idol_emp` (`emp_id` INT NOT NULL , `emp_name` TEXT NOT NULL , `emp_photo` BLOB NOT NULL , `emp_biodata` BLOB NOT NULL , PRIMARY KEY (`id`))

In query (i), the emp_photo and emp_biodata, these two fields require the BLOB data. So their data types are BLOB.

ii) **Data Insertion Query:** As BLOB is MySQL datatype and it has the following four BLOB data type depending on the length of the data that they can hold.

a) TINY BLOB
b) BLOB
c) MEDIUMBLOB
d) LONG BOB

To insert the data into 'idol_emp' using BLOB and python, you need to perform the following steps.

a) You need to install MySQL-Python connector using pip and then need to establish the connection.
b) You need a python function that converts images and other multimedia data into binary data.
c) Then define your insert query and execute the query using the cursor.execute() function.
d) After the query execution, you need to commit your database changes.
e) Then you need to close your cursor and database connection.
f) Finally, verify your result.

The code of insertion into the database using the BLOB is given below.

Space for learners:

```
import mysql.connector
def multimediaToBinary(filename):
    with open(filename, 'rb') as file:
        binaryData = file.read()
    return binaryData
def insertBLOB(emp_id, emp_name, emp_photo, emp_biodata):
    print("Inserting multimedia data into idol_emp")
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='idol_db',
                                             user='idol',
                                             password='idolidol')

        cursor = connection.cursor()
        sql_insert_blob = """ INSERT INTO idol_emp
                          (emp_id, emp_name, emp_photo, emp_biodata)
VALUES (%s,%s,%s,%s)"""
        emp_photo = convertToBinaryData(emp_photo)
        emp_biodata = convertToBinaryData(emp_biodata)
        insert_blob = (emp_id, emp_name, emp_photo,
emp_biodata)
        result = cursor.execute(sql_insert_blob, insert_blob)
        connection.commit()
        print("Image and biodata has inserted successfully ", result)

    except mysql.connector.Error as error:
        print("Failed inserting multimedia data {}".format(error))

    finally:
        if connection.is_connected():
            cursor.close()
```

Space for learners:

```
        connection.close()
        print("MySQL connection is closed")
insertBLOB(1, "idol_emp1", "path of the image",
        "path of the text")
```

After data insertion in a database using the BLOB in MySQL, you can retrieve the data from the database as given below. For the same, MySQL and python connector is required as stated above. But to execute the select query cursor.execute() function is required. Then you can use cursor.fetchall() to retrieve the data from the database as below.

```
import mysql.connector

def write_file(data, filename):
Disk
   with open(filename, 'wb') as file:
      file.write(data)

def readBLOB(emp_id, emp_photo, emp_bioData):
   print("Reading data from idol_emp table")
   try:
      connection = mysql.connector.connect(host='localhost',
                                    database='idol_db',
                                    user='idol',
                                    password='idolidol')
      cursor = connection.cursor()
      sql_fetch = """SELECT * from idol_emp where id = %s"""
      cursor.execute(sql_fetch, (emp_id,))
      record = cursor.fetchall()
      for row in record:
         print("Employee Id = ", row[0], )
         print("Employee Name = ", row[1])
         Employee image = row[2]
         Employee biodata = row[3]
```

Space for learners:

```
        print("Storing employee's photyo and biodata in the
local PC")
        write_file(Employee image, emp_photo)
        write_file(Employee biodata, emp_biodata)
    except mysql.connector.Error as error:
        print("Failed to read data from idol_emp {}".format(error))
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("MySQL connection is closed")
readBLOB(1, "path of the image",
```

Space for learners:

CHECK YOUR PROGRESS-I

1. What do you mean by multimedia database?
2. What are the BLOB data types?
3. State truth or false
   i. TINY BLOB is Blob data type
   ii. MEDIUMBLOB is not a blob data type.
   iii. Images consist of pixel
4. What is cursor.execute() function?
5. What is the role of cursor.fetchall() ?

3.5 REQUIREMENT OF MULTIMEDIA DATABASE

Like other DBMS, the multimedia database should address the requirement issues.

i) Integration: It indicates that data of a multimedia database should not be duplicate.

ii) Concurrency control: Like other DBMS, a multimedia database should control the concurrency of the transaction. Otherwise, consistency issues will be arises.

iii) Data Independency: In the multimedia database, data of the different multimedia should be independent. It should be managed from the user side.

iv) Persistence: Data of a multimedia database should be saved and reused by the other transactions.

v) Recovery: Data should be recovered at the time of failure. A system may fail due to different reasons, but the recovery option of a multimedia database should recover the data at the time of need.

Space for learners:

3.6 CHALLENGES OF MULTIMEDIA DATABASE

Like other DBMS, multimedia databases also have some challenges. They are presented below.

i) The designing of the multimedia database is not so easy as the different format of data is present in the multimedia database.

ii) As the multimedia database consists of images, text, video, mp3, etc. So conversion of one file format to another format is not so easy.

iii) Storing multimedia data requires more amounts of space. Designing a large dataset is not so easy.

iv) Processing is another issue of multimedia databases because the processing of data requires more amount of time.

v) Multimedia query processing and execution is another issue of the multimedia database.

Space for learners:

CHECK YOUR PROGRESS-II

6. What is concurrency in multimedia database?
7. State truth or false
   i. Integration is a requirement of multimedia database.
   ii. Data independecy should be a part of multimedia database.
   iii. All multimedia shoud not be recovered.
8. State two challenges of multimedia database.

3.7 CONTENTS OF MULTIMEDIA DATABASE

The multimedia database store the multimedia information. It contains the following information.

i) The multimedia database contains the multimedia data like audio, video, text, animations, and images.

ii) The media of the multimedia information contains the sampling rate, the frame rate of the data signal.

iii) The keyword data is used to represent the data of a multimedia database such as image keyword means its date, time, and description of the image.

iv) The media feature data contains the features of a data. For example, an image means its color, texture, and shape.

3.8 APPLICATION OF MULTIMEDIA DATABASE

The multimedia database can be applied in the following areas.

i) The Multimedia databases can be applied in the area of document and record management systems such as insurance claim records.

ii) Multimedia databases can be applied in digital libraries. For example IR@ inflibnet.

iii) The Multimedia databases are used in the video on demand. For example. Netflix

iv) A Multimedia database is used in music. For example. Ganna.

v) The multimedia database is used in GIS. For example. Landsat 8.

CHECK YOUR PROGRESS-III

9. State truth or false
   i. Audio is not a part of multimedia database.
   ii. Ganna.com is an example of multimedia database.
10. State two applications multimedia database.
11. What is a netflex?
12. What is inflibnet?

3.9 SUMMING UP

- A Multimedia database is a collection of multimedia data such as texts, images, videos, audios, etc.
- An image is multimedia data. It consists of the pixel. The pixel of an image contains all the necessary information about the image.
- An image may be color, grayscale, or black and white.
- The embedded SQL is used to insert the images into a database. A database should support an image to insert an image in the database.
- The images are stored in binary form in the cell of a table of a database and the data type of the cell is Binary Large Object (BLOB). It is a MySQL data type that is not only used to store the image data but also used to store the other data type.

Space for learners:

- **Table Creation Query:** CREATE TABLE `idol_emp` (`emp_id` INT NOT NULL , `emp_name` TEXT NOT NULL , `emp_photo` BLOB NOT NULL , `emp_biodata` BLOB NOT NULL , PRIMARY KEY (`id`))
- **Data Insertion Query:** As BLOB is MySQL datatype and it has the following four BLOB data type depending on the length of the data that they can hold.
  a) TINY BLOB
  b) BLOB
  c) MEDIUMBLOB
  d) LONG BOB
- Like other DBMS, the multimedia database should address the requirement issues.
  1. Integration:
  2. Concurrency control
  3. Data Independency
  4. Persistence
  5. Recovery
- Like other DBMS, multimedia databases also have some challenges. They are.
  1. Designing of Multimed a database.
  2. One File Format for multimedia data.
  3. Processing is another issue of multimedia databases because the
  4. Multimedia query processing and execution.
- The multimedia database store the multimedia information. It contains the following information.
  1. Audio, video, text, animations, and images.
  2. The sampling rate, the frame rate of the data signal.
  3. Image keyword means its date, time, and description of the image.
  4. Features of data.

Space for learners:

- The multimedia database can be applied in the following areas.
  1. Insurance claim records.
  2. Inflibnet.
  3. Netflix
  4. Ganna.
  5. Landsat 8

Space for learners:

3.10 ANSWERS TO CHECK YOUR PROGRESS

1. A Multimedia database is a collection of multimedia data such as texts, images, videos, audios, etc.
2. The four BLOB data types are
   i. TINY BLOB
   ii. BLOB
   iii. MEDIUMBLOB
   iv. LONG BOB
3. i) True ii) False iii) True
4. To insert data in the multimedia database, the insert queries are executed using the cursor.execute() function.
5. Using the cursor.fetchall(), one can retrieve the data from the multimedia database.
6. In a database management system (DBMS), concurrency control manages simultaneous access to a database. Like other DBMS, a multimedia database should control the concurrency of the transaction. Otherwise, consistency issues will be arises.
7. i) True ii) True iii) False
8. Like other DBMS, multimedia databases also have some challenges. They are presented below.
   i. The designing of the multimedia database is not so easy as the different format of data is present in the multimedia database.

ii. As the multimedia database consists of images, text, video, mp3, etc. So conversion of one file format to another format is not so easy

9. i) False ii) True
10. The multimedia database can be applied in the following areas.
    i. Multimedia databases can be applied in digital libraries. For example IR@ inflibnet.
    ii. The Multimedia databases are used in the video on demand. For example. Netflix
11. Netflix is an example of a multimedia database. It is a streaming service that offers a wide variety of award-winning TV shows, movies, anime, documentaries, etc.
12. Information and Library Network (INFLIBNET) is an example of a multimedia database and is an autonomous Inter-University Centre of the University Grants Commission (UGC) that provides access to e-resources to colleges, universities, and centrally funded technical institutions

Space for learners:

3.11 POSSIBLE QUESTIONS

Short answer type questions:

1. What is a multimedia database?
2. Define the terms image, video, and audio.
3. What is BLOB?
4. Why do you need BLOB?
5. What are the data types of BLOB?
6. What is an embedded query?
7. How do create a multimedia table?
8. How do you insert queries in a multimedia table?
9. State two issues of a multimedia database?
10. State two design issues of the multimedia database.
11. State two challenges of the multimedia database.
12. State two applications of a multimedia database.

Long answer type questions:

1. Explain the data insertion and retrieve in a multimedia database using python.
2. Explain the different challenges of a multimedia database.
3. Explain the requirements of a multimedia database.

Space for learners:

3.12 REFERENCES AND SUGGESTED READINGS

1. Multimedia Database Management Systems, Author: Prabhakaran, Publisher: Springer
2. Multimedia Database Management Systems, Author: Guojun Li.

UNIT 4: SPATIAL DATABASE

Unit Structure:

Space for learners:

4.1 INTRODUCTION

This unit gives an overview of the spatial database. A spatial database is one in which the geographic location information is saved. The concept of a spatial database is explained here along with its indexing techniques. Content-based indexing is also discussed in this unit and along with the retrieval techniques. Content-based image indexing means the properties of an image are saved in the database and it is retrieved based on its properties. Finally, the different indexing techniques such as R trees, R+ Tress, and KD tree is discussed in the unit.

4.2 UNIT OBJECTIVES

After learning this unit, you will be able to learn

i) About the concept spatial database

ii) About the process of saving location in database

iii) About the Content-Based Indexing (CBI) and its retrieving.

iv) About the indexing technique such as R tree, R+ tree, and KD tree.

Space for learners:

4.3 SPATIAL DATABASE CONCEPT

The spatial data is one where the geographic location such as a village, town. Cities or locations are associated. The spatial database is where you can be saved this type of information in terms of some location object data. Technically, a spatial database is optimized for storing and querying object data in a geometric space. The Spatial database stores geometric objects like points, lines, and polygons but some databases are also saved 2d objects, linear networks, etc. It is a part of the GIS database (Fig.3.1)

Let's understand the concept of a spatial database with the help of the following example. Let's have a satellite image of a road. Though it is an image it has geographic information such as points, lines, and polygons which represent the building, rod, etc. So, this image, the spatial data is represented by vector data and raster data. You can say that spatial data are two types.

i) Vector data: The data is represented using lines, points, and polygons.

ii) Raster data: The data is presented using the matrix. For example, data for building.

The database system which manages the spatial data is known as Spatial Data Base Management System (SDBMS). The SDBMS plays a prominent role in the management of queries of spatial data. Spatial data is used in many disciplines such as geography, remote sensing, urban planning, and natural resource management. As mentioned above, the spatial database established a specification to represent the above two types of spatial data. Using this specification, spatial queries are processed using the SQL (For example PostgreSQL. PostGIS, QGIS).

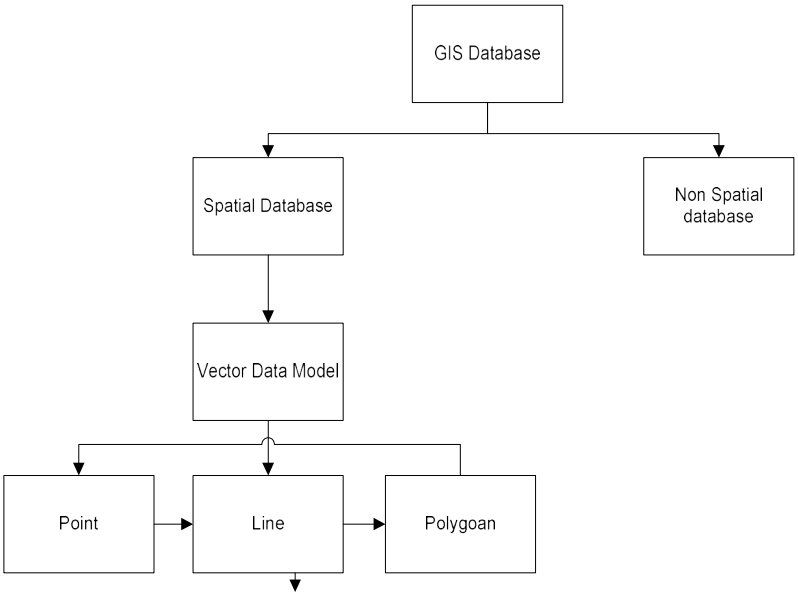


Fig.3.1. Spatial Database Types

Other database uses indexing technique to access the data faster and search the data most efficiently. But these direct indexing techniques may not work properly in the case of spatial databases. So, it needs a special type of indexing known as content indexing. The spatial indexes such as R tree, R+tree, etc. are designed for spatial indexing. It is required to retrieve spatial data from a large database. Apart from indexing, spatial databases offer spatial data types in their data model and query language. This special data type is required to model the spatial database.

A spatial database can be applied in many areas. A few of the applications are presented below.

i) **Image and Multimedia databases:** In the multimedia database, the spatial database is applied such as content-based image retrieval, content-based video retrieval, medical database, etc.

ii) **Time-series databases**: In the management of time intervals, the spatial database is used.

iii) **Traditional DBMS:** In the case of data warehouses, the SDBMS is used.

iv) **Socio-Economic applications:** In Urban planning, the Route optimization problem, and the market analysis of the SDBMS are used.

Space for learners:

v) **Environmental applications:** In the case of Fire or Pollution Monitoring, the SDBMS is used

vi) **Administrative applications**: In Public networks administration and vehicle navigation, the SBMS is used.

The SDBMS are necessary for the following requirements before its designs.

i) For the manipulation of very large amounts of data, e.g., terabytes of data per day from satellite images, the SDBMS is required.

ii) For data distinction, e.g., spatial and non-spatial (alphanumeric) data, the DBMS is necessary.

iii) For Complex spatial relationships and operations, e.g., topological, directional, metric relationships, the SDBMS is necessary.

iv) Complex spatial relationships, e.g., find all cities adjacent to a river, find all dark shapes left to the heart, and find the 5 closest hospitals concerning a given location.

v) Spatial join: An expensive operation, e.g., Find the 5 closest hospitals concerning any highway.

CHECK YOUR PROGRESS-I

1. What do you mean by spatial data and spatial database?
2. State few applications of spatial database.
3. Which classes does spatial data types in MySQL correspond to?
4. Stet true or false
   i) SPATIAL indexes cannot be created on NOT NULL spatial columns.
   ii) By *'spatial data'* we mean data that has position value.

4.4 SPATIAL DBMS DATA MODELS

In section 4.3, the two types of the data model of SDBMS are already mentioned. They are

i) **Raster Model**: In the raster model, SDBMS spaces are subdivided into cells of regular size and shape such as square,

Space for learners:

triangle, hexagon, etc. Each cell of the raster is assigned the value of the attribute it represents and only one value is assigned for the same. Different attributes are stored in separate files (layers).

ii) **Vector Model**: In the vector model, the subdivision of the space is done based on the position of the geographic feature, i.e., irregular. The features are represented by (2-D space), such as Points (x,y), Lines (x1,y1, x2,y2, ..., xn,yn), Regions (x1,y1, ..., xn,yn, x1,y1).

Space for learners:

4.5 CONTENT-BASED INDEXING AND RETRIEVAL

Like another database system, SDBMS also needs indexing of spatial data for faster query processing and searching spatial data most efficiently. Content-based indexing is one where data is saved based on the properties of the data and it is retrieved based on these properties. "Content-based" means that the search analyses the contents of the data rather than the metadata such as keywords, tags, or descriptions associated with the data. It results in faster query processing and searching. For example, the Content-Based Image Indexing and Retrieval (CBIR) system are where images are saved in the database based on the properties of the image data. The properties of the image mean its color, texture, and shape. So based on these, you can index the images and retrieve also.

The content indexing and retrieving are applicable in many areas but image, video, text, music are very popular ones. Let's explain the CBIR with a help of an example. Let's you have many images in your database. In CBIR, query images will be there with you. Initially, the feature of the image such as color, texture, position, shape, etc can extract from the query image. The features are saved as a vector for the query image and the same process can be applied to the database also. When you have both the feature vector, just compare the feature vector of the query image with the feature vector of the database image.

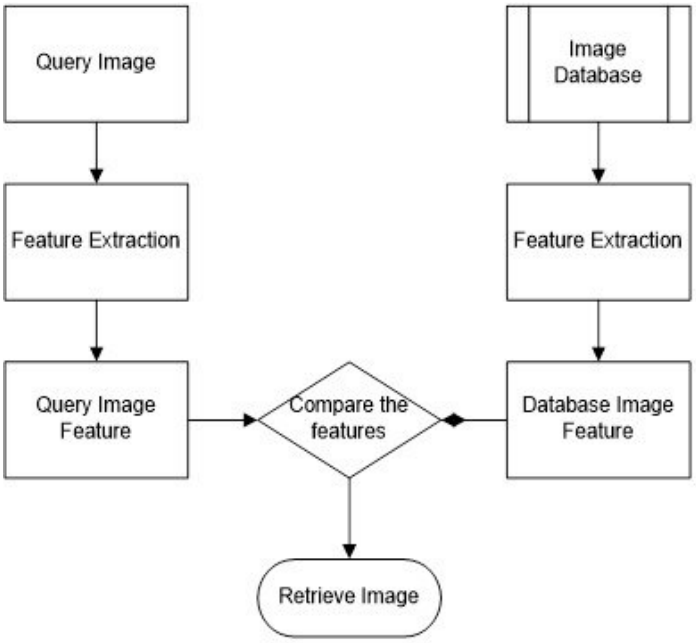


Fig. 3.2: CBIR system

CHECK YOUR PROGRESS-II

5. What are the spatial data models?
6. Can we use image and video for CBIR system?
7. Give two examples of CBIR search engine.
8. What features of an image are considered for CBIR?

4.6 DIFFERENT INDEXING TECHNIQUES

For the indexing of spatial data, different indexing techniques are used.

i) R Tree
ii) R+ Tree
iii) KD Tree

Let's explain these techniques with the help of examples.

I. **R tree:** R-tree is a tree data structure to store the spatial data efficiently. It is used for storing spatial data indexes. It is useful for spatial data queries, storage, and indexing. are highly useful for spatial data queries and storage. Indexing multi-dimensional information. For example, handling of game

Space for learners:

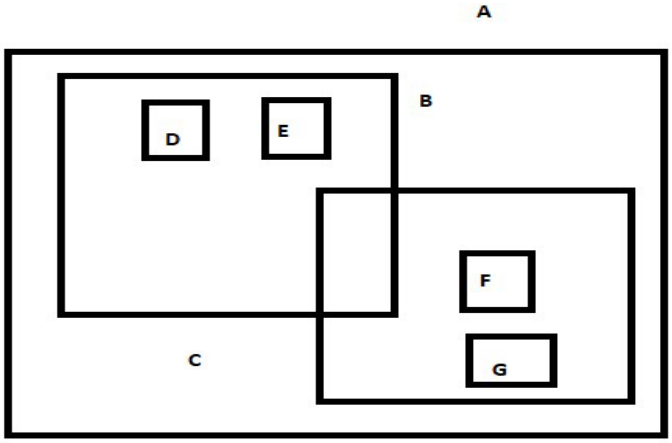
data, virtual maps implementation, and handling geospatial coordinates, etc. The properties R tree are given below.

i. Consists of a single root with internal and leaf nodes.

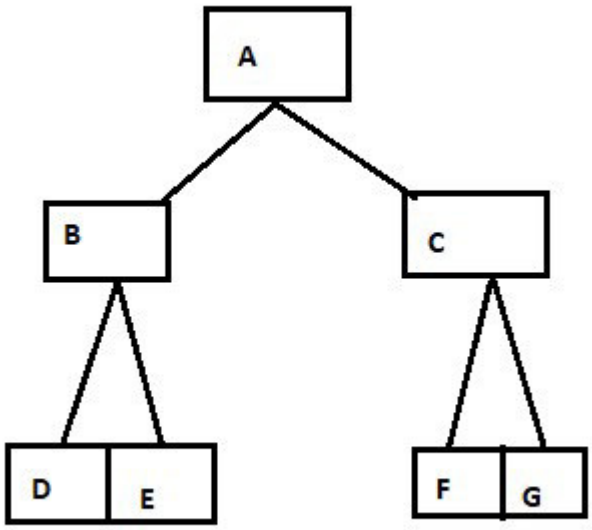
ii. The root node contains a pointer to the largest region.

iii. The parent nodes contain pointers to their child nodes where the region of child nodes completely overlaps with the regions of parent nodes.

iv. Leaf nodes contain the actual data within the Minimum Bounding region (MBR) to the current objects where the MBR is the sub-regions within the entire space that group data as efficiently.



(a)



Space for learners:

(b)

Fig. 3.3 R tree in SDBMS

To locate an object, the search algorithm descends the tree from the root. The algorithm recursively traverses down the subtrees of bounding rectangles that intersect the query rectangle. When a leaf node is reached, bounding rectangles are tested against the query rectangle and their objects are fetched for testing if they intersect the query rectangle.

II. **R +Tree**: R+tree is a variant of R trees where data is indexed using (x,y) coordinates. It is a conciliation between the R tree and the KD tree. In the R+ tree, the nodes may not be half-filled and the internal nodes of the tree avoid overlapping by inserting an object into multiple leaves. In the R+ tree, the tree has minimal coverage and minimal overlap and it overcomes the overlapping issue of the R tree. The advantages and disadvantages of the R+ tree are presented below.

   Advantages:

   i) Due to no overlapped between the nodes, the point query performance benefits are covered by at most one node.

   ii) A single path is identified to visit the nodes.

   Disadvantages:

   i) Since rectangles are duplicated, an R+ tree can be larger than an R tree built on the same data set.

   ii) Construction and maintenance of R+ trees are more complex than the R trees and other variants of the R tree.

The duplication of objects or nodes in R+tree leads to the non-overlapping of entries. If the corresponding covering rectangles intersect the query region, then only the searching is possible in R+tree. The disjoint covering rectangles avoid the multiple search paths of the R-tree for point queries.

To insert an object, multiple paths may be traversed. At a node, the subtrees with covering rectangles that intersect with the object bounding rectangle must be traversed. On reaching the leaf nodes, the object identifier will be stored in the leaf nodes.

Space for learners:

Multiple noes of R+tree may store the same object. Three cases should take care of the insertion.

i) Insert an object into a node where the covering rectangles of all entries do not intersect with the object-bounding rectangle.

ii) The second one is when the bounding rectangle of the new object only partially intersects with the bounding rectangles of entries.

iii) The third case is more serious in that the covering rectangles of some entries can prevent each other from expanding to include the new object.

III. **KD Tree**: KD tree is a binary search tree that is also known as K dimensional tree. In the KD tree, the data in each node of the tree represents the K dimensional point in space. So it is also known as space partitioning data structure. It represents the points or data in K dimensional space. The non-leaf node of the KD tree effectively divides the tree into two spaces, known as half-space. The data that is left of the root will go into a left sub-tree, data right of a root will go in a right subtree. Construction of the KD tree is as follows:

i) The axis used to generate splitting trees is cycled repeatedly.

ii) The nodes are selected by taking the median of the data being placed in the subtree.

Let's understand the basic concept of the KD tree by considering a 2D tree. In the KD tree, the left subtree contains those points whose coordinates are smaller than the root node, and the right subtree contains those points whose coordinates are grater-equal to the root node.

Let’s build a k-d tree with the points: (30,40), (5,25), (70,70), (50,30), (35,45). Let the root node is x aligned.

i) Take the first coordinates (30,40). As the tree is empty, so make it the root node of the tree.

ii) Now the 2nd coordinate is (5,25). As the first x value of the 2nd coordinates is 5 and 5<30. So it will go to the left subtree.

Space for learners:

iii) The 3rd coordinate is (70,70). Now 70>30, so it will go right subtree.

iv) The 4th coordinate is (50,30). First, compare with root 50>30. But already (70,70) is in the right subtree. Now Compare 50 with the y value of 70. 50<70. So, it will be in the left subtree of (70,70).

v) The final coordinate is (35,45). Comparing with root (35>30). It will go right subtree. In the right subtree, comparing with y coordinates of (70,70), you find 35<70. So, it will go left subtree of (70,70). But in the left subtree of (70,70), the (50,30) coordinate is present. Now compare 35 of (35,45) with x of (50,30). So, 35<45. So, it will be on the left of (50,30).

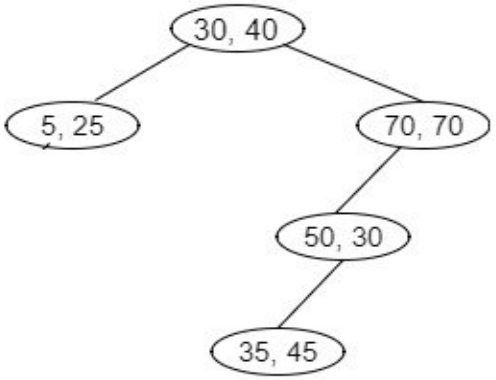


Fig. 3.4 KD tree in SDBMS

CHECK YOUR PROGRESS-III

9. What is r tree and r+ tree?
10. In what time can a 2-d tree be constructed?
11. In a k-d tree, what is K meant?
12. Each level in a k-d tree is made of cutting and dimension (True or False)

Space for learners:

4.7 SUMMING UP

Space for learners:

- The spatial data is one where the geographic location such as a village, town. Cities or locations are associated. The spatial database is where you can be saved this type of information in terms of some location object data.
- The spatial data are two types.
  - Vector data: The data is represented using lines, points, and polygons.
  - Raster data: The data is presented using the matrix. For example, data for building.
- The spatial queries are processed using the SQL (For example PostgreSQL. PostGIS, QGIS).
- The SDBMS are necessary for the following requirements before its designs.
  - For the manipulation of very large amounts of data.
  - For data distinction.
  - For Complex spatial relationships and operations.
  - Complex spatial relationships.
  - Spatial join.
- Content-Based Image Indexing and Retrieval (CBIR) system are where images are saved in the database based on the properties of the image data.
- R-tree is a tree data structure to store the spatial data efficiently. It is used for storing spatial data indexes. It is useful for spatial data queries, storage, and indexing. are highly useful for spatial data queries and storage. Indexing multi-dimensional information.
- R+tree is a variant of R trees where data is indexed using (x,y) coordinates. It is a conciliation between the R tree and the KD tree.
- KD tree is a binary search tree that is also known as K dimensional tree. In the KD tree, the data in each node of the tree represents the K dimensional point in space.
- Construction of the KD tree is as follows:

- The axis used to generate splitting trees is cycled repeatedly.
- The nodes are selected by taking the median of the data being placed in the subtree.

Space for learners:

4.8 ANSWERS TO CHECK YOUR PROGRESS

1. The spatial data is one where the geographic location such as a village, town. Cities or locations are associated. The spatial database is where you can be saved this type of information in terms of some location object data.
2. The few applications of spatial DBMS are
   i) Image and Multimedia databases
   ii) Time-series databases
   iii) Traditional DBMS
3. OpenGIS
4. i) FALSE
   ii) TRUE
5. Two types of spatial data models are raster and vector model.
6. For CBIR, we can use only image.
7. eBay image Search and Google Image Search.
8. Color, shape, and texture.
9. R and R+ tree are spatial indexing techniques in spatial DBMS.
10. O(nlogn)
11. Number of dimensions.
12. True

4.9 POSSIBLE QUESTIONS

Short answer type questions:

1) What do you mean by spatial data and spatial database system?

2) Explain raster and vector model in spatial DBMS.
3) What is CBIR? Give examples.
4) What are the applications of spatial DBMS?
5) What are the requirements of Spatial SBMS?
6) State the difference between R and R+ tree.
7) Give some examples of spatial query language.
8) What are advantages of R+ tree over R tree?
9) Why do need KD tree if you have R and R+ tree?
10) What are the requirements of spatial DBMS?

Long answer type questions:

1) Explain the CBIR system with an example and diagrams.
2) Explain the KD tree with an example.
3) What are the indexing techniques of Spatial DBMS? Explain.

Space for learners:

4.10 REFERENCES AND SUGGESTED READINGS

1) Spatial Databases: With Application to GIS, by Michel O. Scholl
2) Spatial Data Management by Nikos Mamoulis