ADVANCED DATABASE SYSTEM

Prepared by

Mrs. M.Gayathri

Assistant Professor

Dept of Computer Science

Advanced Database System

Unit I:

Database System: Introduction- Overview of Database Management Systems- Data Independence- Database System Architecture-The External level- The Conceptual level- The Internal Level—Mappings-The DBA- Data Dictionary-Data Models- Record Based Data Models-Object Based Data Models-Physical Data Models-Hierarchical Data Models-Network Data Models-Relational Data Models-E-R Models- Object Oriented Data Model

Unit II

Distributed Databases and Decision Support: The Objectives and Problems of Distributed Data bases-Client Server Systems-DBMS Independence-SQL facilities-Decision Support-Data Preparation-Data Warehouses and Data Marts-OLAP-Object Oriented Data bases: Object Oriented Data Models-Object Oriented Data base-Object Oriented DBMS-Object Oriented Languages

Unit III

Temporal Data bases: Introduction-Packing and Unpacking Relations-Generalizing the Relational Operators-Database Design-Integrity Constraints-Multimedia Databases: Multimedia Sources-Multimedia Database Queries-Multimedia Database Applications

Unit IV

Spatial Databases: Spatial Data-Spatial Database Characteristics-Spatial Data Model-Spatial Database Queries-Techniques of Spatial Database Query-Logic Based Databases: Introduction-Overview-Proportional Calculus-Predicate Calculus-Deductive Database Systems-Recursive Query Processing

Unit V

Emerging Database Technologies: Introduction-Internet Databases-Multimedia Databases-Mobile Databases-MySQL: Introduction-An Overview of MYSQL-MYSQL Database

UNIT-I

INTRODUCTION:

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of data. Meaningful data combined to form information. Hence, information is interpreted data - data provided with semantics. MS. ACCESS is one of the most common examples of database management software.

OVERVIEW

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information.

Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Database Architecture

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics

- Real-world entity − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- Relation-based tables − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- Isolation of data and application − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- Less redundancy − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- Consistency − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- Query Language − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- ACID Properties − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- Multiuser and Concurrent Access − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- Multiple views − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

- Security − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

Data Independence:

Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data. Application programs should not, ideally, be exposed to details of data representation and storage. The DBMS provides an abstract view of the data that hides such details.

There are two types of data independence: physical and logical data independence.

The data independence and operation independence together gives the feature of data abstraction. There are two levels of data independence

Data Independence Types

The ability to modify schema definition in one level without affecting schema definition in the next higher level is called data independence. There are two levels of data independence; they are Physical data independence and Logical data independence.

1. Physical data independence is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance. It means we change the physical storage/level without affecting the conceptual or external view of the data. The new changes are absorbed by mapping techniques.

2. Logical data independence is the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system). Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not change. For example: consider two users A & B. Both are selecting the fields "EmployeeNumber" and "EmployeeName". If user B adds a new column (e.g. salary) to his table, it will not affect the external view for user A, though the internal schema of the database has been changed for both users A & B.

Logical data independence is more difficult to achieve than physical data independence, since application programs are heavily dependent on the logical structure of the data that they access.

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

Database (Data) Tier − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

This architecture has three levels:

1. External level
2. Conceptual level
3. Internal level

1. External level

It is also called **view level**. The reason this level is called “view” is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping.

The user doesn't need to know the database schema details such as data structure, table definition etc. user is only concerned about data which is what returned back to the view level after it has been fetched from database (present at the internal level).

External level is the "**top level**" of the Three Level DBMS Architecture.

2. Conceptual level

It is also called **logical level**. The whole design of the database such as relationship among data, schema of data etc. are described in this level.

Database constraints and security are also implemented in this level of architecture. This level is maintained by DBA (database administrator).

3. Internal level

This level is also known as physical level. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. This is the lowest level of the architecture.

Mappings

Process of transforming request and results between three level it's called mapping.

There are the two types of mappings:

1. Conceptual/Internal Mapping
2. External/Conceptual Mapping

1. Conceptual/Internal Mapping:

- The conceptual/internal mapping defines the correspondence between the conceptual view and the store database.
- It specifies how conceptual record and fields are represented at the internal level.
- It relates conceptual schema with internal schema.
- If structure of the store database is changed.

- If changed is made to the storage structure definition-then the conceptual/internal mapping must be changed accordingly, so that the conceptual schema can remain invariant.
- There could be one mapping between conceptual and internal levels.

2. External/Conceptual Mapping:

- The external/conceptual mapping defines the correspondence between a particular external view and conceptual view.
- It relates each external schema with conceptual schema.
- The differences that can exist between these two levels are analogous to those that can exist between the conceptual view and the stored database.
- Example: fields can have different data types; fields and record name can be changed; several conceptual fields can be combined into a single external field.
- Any number of external views can exist at the same time; any number of users can share a given external view: different external views can overlap.
- There could be several mapping between external and conceptual levels.

Data Base Administrator

Database administration is the function of managing and maintaining database management systems (DBMS) software. Mainstream DBMS software such as Oracle, IBM DB2 and Microsoft SQL Server need ongoing management. As such, corporations that use DBMS software often hire specialized information technology personnel called database administrators or DBAs.

Responsilities

- Installation, configuration and upgrading of Database server software and related products.
- Evaluate Database features and Database related products.
- Establish and maintain sound backup and recovery policies and procedures.

- Take care of the Database design and implementation.
- Implement and maintain database security (create and maintain users and roles, assign privileges).
- Database tuning and performance monitoring.
- Application tuning and performance monitoring.
- Setup and maintain documentation and standards.
- Plan growth and changes (capacity planning).
- Work as part of a team and provide 24x7 support when required.
- Do general technical troubleshooting and give cons.
- Database recovery.

Types

There are three types of DBAs:

1. Systems DBAs (also referred to as physical DBAs, operations DBAs or production Support DBAs): focus on the physical aspects of database administration such as DBMS installation, configuration, patching, upgrades, backups, restores, refreshes, performance optimization, maintenance and disaster recovery.
2. Development DBAs: focus on the logical and development aspects of database administration such as data model design and maintenance, DDL (data definition language) generation, SQL writing and tuning, coding stored procedures, collaborating with developers to help choose the most appropriate DBMS feature/functionality and other pre-production activities.
3. Application DBAs: usually found in organizations those have purchased 3rd party application software such as ERP (enterprise resource planning) and CRM (customer relationship management) systems. Examples of such application software include Oracle Applications, Siebel and PeopleSoft (both now part of Oracle Corp.) and SAP. Application DBAs straddle the fence between the DBMS and the application software and are responsible for ensuring that the application is fully optimized for the

database and vice versa. They usually manage all the application components that interact with the database and carry out activities such as application installation and patching, application upgrades, database cloning, building and running data cleanup routines, data load process management, etc

Data Dictionary

A **data dictionary**, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". Oracle defines it as a collection of tables with metadata. The term can have one of several closely related meanings pertaining to databases and database management systems (DBMS):

- A document describing a database or collection of databases
- An integral component of a DBMS that is required to determine its structure
- A piece of middleware that extends or supplants the native data dictionary of a DBMS
- If a data dictionary system is used only by the designers, users, and administrators and not by the DBMS Software, it is called a passive data dictionary. Otherwise, it is called an active data dictionary or data dictionary. When a passive data dictionary is updated, it is done so manually and independently from any changes to a DBMS (database) structure. With an active data dictionary, the dictionary is updated first and changes occur in the DBMS automatically as a result.
- Database users and application developers can benefit from an authoritative data dictionary document that catalogs the organization, contents, and conventions of one or more databases. This typically includes the names and descriptions of various tables (records or Entities) and their contents (fields) plus additional details, like the type and length of each data element. Another important piece of information that a data dictionary can provide is the relationship between Tables. This is sometimes referred to in Entity-Relationship diagrams, or if using Set descriptors, identifying which Sets database Tables participate in.

- In an active data dictionary constraints may be placed upon the underlying data. For instance, a Range may be imposed on the value of numeric data in a data element (field), or a Record in a Table may be FORCED to participate in a set relationship with another Record-Type. Additionally, a distributed DBMS may have certain location specifics described within its active data dictionary (e.g. where Tables are physically located).
- The data dictionary consists of record types (tables) created in the database by systems generated command files, tailored for each supported back-end DBMS. Oracle has a list of specific views for the "sys" user. This allows users to look up the exact information that is needed. Command files contain SQL Statements for CREATE TABLE, CREATE UNIQUE INDEX, ALTER TABLE (for referential integrity), etc., using the specific statement required by that type of database.
- There is no universal standard as to the level of detail in such a document.

Data Models

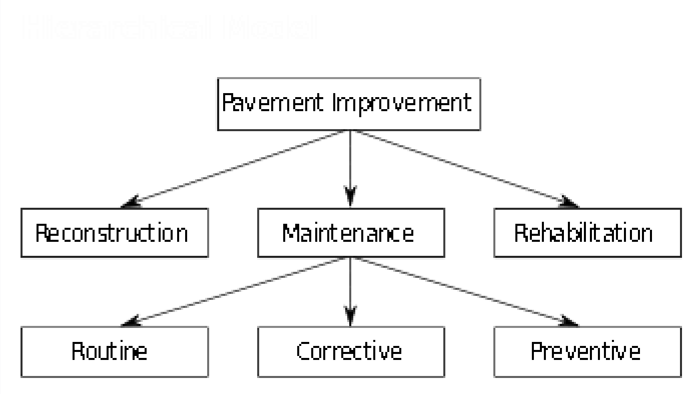
Common logical data models for databases include:

- Hierarchical database model

It is the oldest form of data base model. It was developed by IBM for IMS (information Management System). It is a set of organized data in tree structure. DB record is a tree consisting of many groups called segments. It uses one to many relationships. The data access is also predictable.

- Network model
- Relational model
- Entity–relationship model
    - Enhanced entity–relationship model
- Object model

Hierarchical model

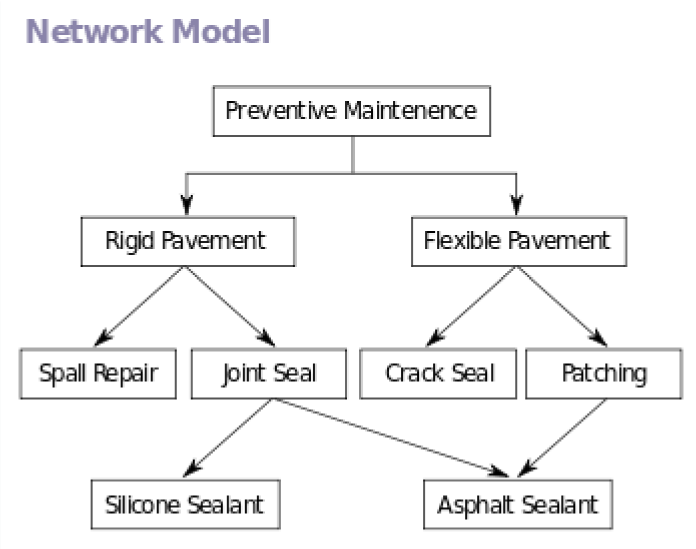


Hierarchical Model

In a hierarchical model, data is organized into a tree-like structure, implying a single parent for each record. A sort field keeps sibling records in a particular order. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM, and now describe the structure of XML documents. This structure allows one one-to-many relationship between two types of data. This structure is very efficient to describe many relationships in the real world; recipes, table of contents, ordering of paragraphs/verses, any nested and sorted information.

This hierarchy is used as the physical order of records in storage. Record access is done by navigating downward through the data structure using pointers combined with sequential accessing. Because of this, the hierarchical structure is inefficient for certain database operations when a full path (as opposed to upward link and sort field) is not also included for each record. Such limitations have been compensated for in later IMS versions by additional logical hierarchies imposed on the base physical hierarchy.

Network model



Network Model

The network model expands upon the hierarchical structure, allowing many-to-many relationships in a tree-like structure that allows multiple parents. It was most popular before being replaced by the relational model, and is defined by the CODASYL specification.

The network model organizes data using two fundamental concepts, called records and sets. Records contain fields (which may be organized hierarchically, as in the programming language COBOL). Sets (not to be confused with mathematical sets) define one-to-many relationships between records: one owner, many members. A record may be an owner in any number of sets, and a member in any number of sets.

A set consists of circular linked lists where one record type, the set owner or parent, appears once in each circle, and a second record type, the subordinate or child, may appear multiple times in each circle. In this way a hierarchy may be established between any two record types, e.g., type A is the owner of B. At the same time another set may be defined where B is the owner of A. Thus all the sets comprise a general directed graph (ownership defines a direction), or network construct. Access to records is either sequential (usually in each record type) or by navigation in the circular linked lists.

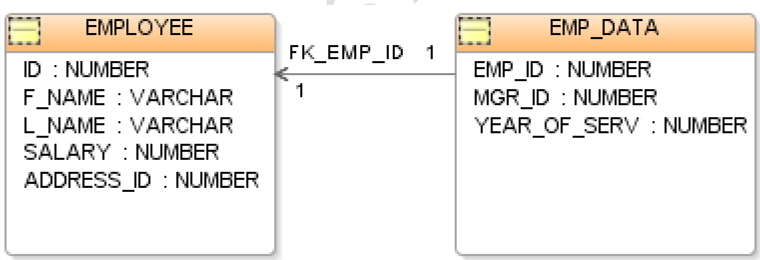
The network model is able to represent redundancy in data more efficiently than in the hierarchical model, and there can be more than one path from an ancestor node to a descendant. The operations of the network model are navigational in style: a program maintains a current position, and navigates from one record to another by following the relationships in which the record participates. Records can also be located by supplying key values.

Although it is not an essential feature of the model, network databases generally implement the set relationships by means of pointers that directly address the location of a record on disk. This gives excellent retrieval performance, at the expense of operations such as database loading and reorganization.

Popular DBMS products that utilized it were Cincom Systems' Total and Cullinet's IDMS. IDMS gained a considerable customer base; in the 1980s, it adopted the relational model and SQL in addition to its original tools and languages.

Most object databases (invented in the 1990s) use the navigational concept to provide fast navigation across networks of objects, generally using object identifiers as "smart" pointers to related objects. Objectivity/DB, for instance, implements named one-to-one, one-to-many, many-to-one, and many-to-many named relationships that can cross databases. Many object databases also support SQL, combining the strengths of both models.

Relational model



Two tables with a relationship

The relational model was introduced by E.F. Codd in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory, and implementations of it have been used by mainframe, midrange and microcomputer systems.

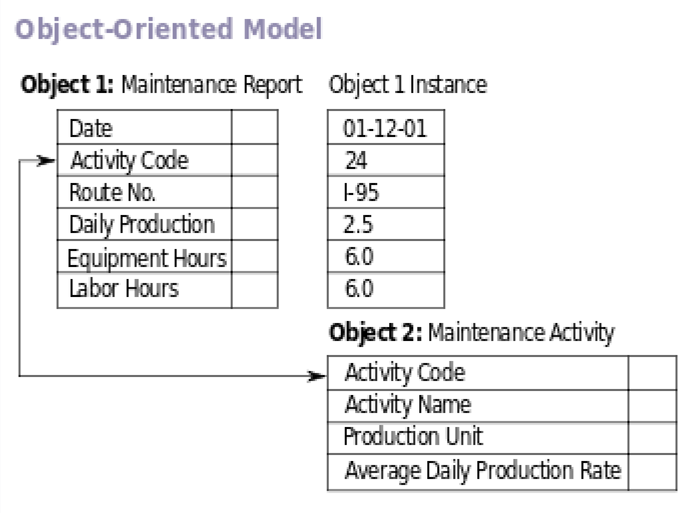
The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models: *relations*, attributes, and *domains*. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

The basic data structure of the relational model is the table, where information about a particular entity (say, an employee) is represented in rows (also called tuples) and columns. Thus, the "relation" in "relational database" refers to the various tables in the database; a relation is a set of tuples. The columns enumerate the various attributes of the entity (the employee's name, address or phone number, for example), and a row is an actual instance of the entity (a specific employee) that is represented by the relation. As a result, each tuple of the employee table represents various attributes of a single employee.

All relations (and, thus, tables) in a relational database have to adhere to some basic rules to qualify as relations. First, the ordering of columns is immaterial in a table. Second, there can't be identical tuples or rows in a table. And third, each tuple will contain a single value for each of its attributes.

A relational database contains multiple tables, each similar to the one in the "flat" database model. One of the strengths of the relational model is that, in principle, any value occurring in two different records (belonging to the same table or to different tables), implies a relationship among those two records. Yet, in order to enforce explicit integrity constraints, relationships between records in tables can also be defined explicitly, by identifying or non-identifying parent-child relationships characterized by assigning cardinality (1:1, (0)1:M, M:M). Tables can also have a designated single attribute or a set of attributes that can act as a "key", which can be used to uniquely identify each tuple in the table.

Object-oriented database models



Object-Oriented Model

Object Oriented Model aims to avoid the object-relational impedance mismatch - the overhead of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). Even further, the type system used in a particular application can be defined directly in the database, allowing the database to enforce the same data integrity invariants. Object databases also introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

UNIT II

The Objectives and Problems of Distributed Databases

1. **Local Autonomy**

All operations at a given site are controlled by that site. All operations at a given site are controlled by that site. No site X should depend on some other site Y for its successful operation. No site X should depend on some other site Y for its successful operation. -- Otherwise site Y is down might mean that site X is unable to run even if there is nothing wrong with site X itself. -- Otherwise site Y is down might mean that site X is unable to run even if there is nothing wrong with site X itself.

2. No Reliance on a Central Site

All sites must be treated as equals. All sites must be treated as equals. There must not be any reliance on a central “master” site for some central service—for example, centralized transaction management. There must not be any reliance on a central “master” site for some central service—for example, centralized transaction management. Two reasons:

1. The central site might be a bottleneck.

2. If the central site went down, the whole system would be down.

3. Continuous Operation

Provide greater reliability and greater availability – it is the advantage of distributed systems in general. Provide greater reliability and greater availability – it is the advantage of distributed systems in general. Unplanned shutdowns are undesirable, but hard to prevent entirely. Unplanned shutdowns are undesirable, but hard to prevent entirely. Planned shutdowns should never be required. Planned shutdowns should never be required.

4. Location Independence

Also known as location transparency. Users should not have to know where data is physically stored, but rather should be able to behave as if the data were all stored at their own local site. Users should not have to know where data is physically stored, but rather should be able to behave -- as if the data were all stored at their own local site.

5. Fragmentation Independence

A system supports data fragmentation if a given base relation can be divided into pieces or fragments for physical storage purposes. A system supports data fragmentation if a given base relation can be divided into pieces or fragments for physical storage purposes. Two benefits: Two benefits: 1. most operations are local 1. Most operations are local 2. Reduce network traffic 2. Reduce network traffic

An example of fragmentation Define two fragments: Define two fragments: FRAGMENT EMP AS FRAGMENT EMP AS N_EMP AT SITE ‘New York’ WHERE DEPT# = DEPT#(‘D1’) N_EMP AT SITE ‘New York’ WHERE DEPT# = DEPT#(‘D1’) OR DEPT# = DEPT#(‘D3’) OR DEPT# = DEPT#(‘D3’) S_EMP AT SITE ‘Shanghai’ WHERE DEPT# = DEPT#(‘D2’) S_EMP AT SITE ‘Shanghai’ WHERE DEPT# = DEPT#(‘D2’) EMP#DEPT#SALARY E1D140K E2D142K E3D230K E4D235K E5D348K User perception EMPEMP#DEPT#SALARYE1D140K E2D142K E5D348K EMP#DEPT#SALARYE3D230K E4D245K New York N_EMP Shanghai S_EMP

6. Replication Independence

A system supports data replication if a given base relation or fragment can be represented in storage by many distinct copies or replicas, stored at many distinct sites. A system supports data replication if a given base relation or fragment can be represented in storage by many distinct copies or replicas, stored at many distinct sites. Ideally should be “transparent to the user”. Ideally should be “transparent to the user”.

Desirable for two reasons:

1. Applications can operate on local copies instead of remote sites.

2. At least one copy available

7. Distributed Query Processing

A relational distributed system is likely to outperform a nonrelational one by orders of magnitude. The query that involves several sites, there will be many possible ways of moving data around the system. The query that involves several sites, there will be many possible ways of moving data around the system.

8. Distributed Transaction Management Recovery

The system must ensure that the set of agents for that transaction either all commit in unison or all roll back in unison. Achieved by two-phase commit protocol. Concurrency Typically based on locking. Typically based on locking.

9 Hardware Independence

Real world involves a multiplicity of different machines—IBM machines, HP machines, PCs and workstations of various kinds. Need to be able to integrate the data on all of those systems. Desirable to be able to run the same DBMS on different hardware platform.

10. Operating System Independence

Be able to run the same DBMS on different operating system platforms.

11. Network Independence

Desirable to be able to support a variety of disparate communication networks also.

12. DBMS Independence

All needed is that the DBMS instances at different sites all support the same interface– they don’t necessarily all of the same DBMS software. For example, if Ingres and Oracle both supported the official SQL standard, the Ingres site and the Oracle site might be able to talk to each other in a distributed database system.

Client Server Systems

Client/server is a term used to describe a computing model for the development of computerized systems. This model is based on the distribution of functions between two types of independent and autonomous processes; servers and clients.

Basic Components

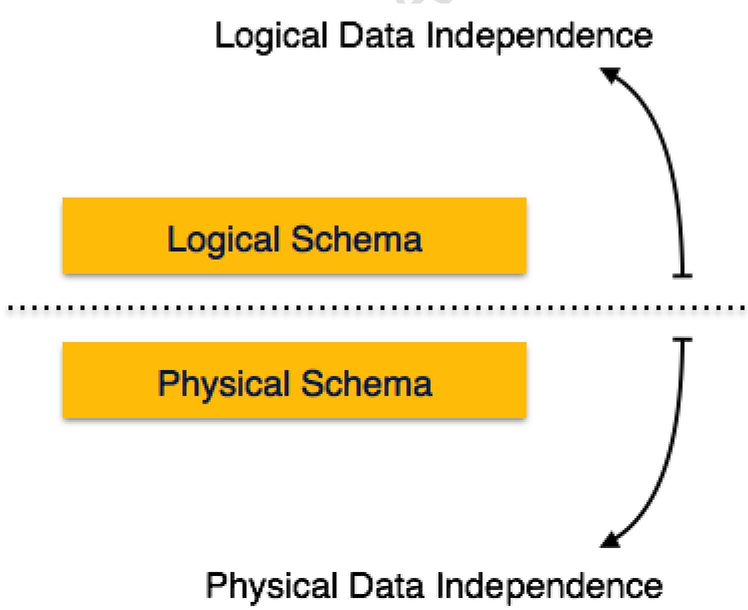
A client is any process that requests specific services from server processes. A server is a process that provides requested services for clients. Both clients and servers can reside in the same computer or in different computers connected by a network.

Variations on Client Server

The key to client/server power is where the requested processing takes place. In mainframe systems and Application Server based systems all processing takes place on the server, and the client is used to display the data screens. With PC and File servers all processing takes place on the PC and the server is used only for storage. There are many variations of these models. The client/server environment provides a clear separation of server and client processes.

DBMS Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation. Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data. For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

Decision Support

A **decision support system** (**DSS**) is an information system that supports business or organizational decision-making activities. DSSs serve the management, operations and planning levels of an organization (usually mid and higher management) and help people make decisions about problems that may be rapidly changing and not easily specified in advance—i.e. unstructured and semi-structured decision problems. Decision support systems can be either fully computerized or human-powered, or a combination of both. While academics have perceived DSS as a tool to support decision making processes, DSS users see DSS as a tool to facilitate organizational processes.[1] Some authors have extended the definition of DSS to include any system that might support decision making and some DSS include a decision-making software component; Sprague (1980) defines a properly termed DSS as follows:

1. DSS tends to be aimed at the less well structured, underspecified problem that upper level managers typically face;
2. DSS attempts to combine the use of models or analytic techniques with traditional data access and retrieval functions;
3. DSS specifically focuses on features which make them easy to use by non-computer-proficient people in an interactive mode; and
4. DSS emphasizes flexibility and adaptability to accommodate changes in the environment and the decision making approach of the user.

DSSs include knowledge-based systems. A properly designed DSS is an interactive software-based system intended to help decision makers compile useful information from a combination of raw data, documents, and personal knowledge, or business models to identify and solve problems and make decisions.

Typical information that a decision support application might gather and present includes:

- inventories of information legacy assets (including and relational data sources, cubes,
- data warehouses, and data marts),
- comparative sales figures between one period and the next,
- projected revenue figures based on product sales assumptions.

Data Preparation

Data preparation is the act of manipulating (or pre-processing) raw data (which may come from disparate data sources) into a form that can readily and accurately be analyzed, e.g. for business purposes. Data preparation is the first step in data analytics projects and can include many discrete tasks such as loading data or data ingestion, data fusion, data cleaning, data augmentation, and data delivery. The issues to be dealt with fall into two main categories:

- systematic errors involving large numbers of data records, probably because they have come from different sources;

individual errors affecting small numbers of data records, probably due to errors in the original data entry.

OLAP

At the core of any OLAP system is an OLAP cube (also called a 'multidimensional cube' or a hypercube). It consists of numeric facts called measures that are categorized by *dimensions*. The measures are placed at the intersections of the hypercube, which is spanned by the dimensions as a vector space. The usual interface to manipulate an OLAP cube is a matrix interface, like Pivot tables in a spreadsheet program, which performs projection operations along the dimensions, such as aggregation or averaging.

The cube metadata is typically created from a star schema or snowflake schema or fact constellation of tables in a relational database. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables. Each *measure* can be thought of as having a set of *labels*, or meta-data associated with it. A *dimension* is what describes these *labels*; it provides information about the *measure*. A simple example would be a cube that contains a store's sales as a *measure*, and Date/Time as a *dimension*. Each Sale has a Date/Time *label* that describes more about that sale.

For example:

```
 Sales Fact Table
+-------------+----------+
| sale_amount | time_id  |
+-------------+----------+          Time Dimension
|      2008.10|     1234 |----+     +---------+-------------------+
+-------------+----------+    |     | time_id | timestamp         |
                              |     +---------+-------------------+
                              +---->|   1234  | 20080902 12:35:43 |
                                    +---------+-------------------+
```

Multidimensional databases

Multidimensional structure is defined as a variation of the relational model that uses multidimensional structures to organize data and express the relationships between data. The structure is broken into cubes and the cubes are able to store and access data within the confines of each cube. Each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions. Even when data is manipulated it remains easy to access and continues to constitute a compact database format. The data still remains interrelated. Multidimensional structure is quite popular for analytical databases that use online analytical processing (OLAP) applications. Analytical databases use these databases because of their ability to deliver answers to complex business queries swiftly. Data can be viewed from different angles, which gives a broader perspective of a problem unlike other models.

Aggregations

It has been claimed that for complex queries OLAP cubes can produce an answer in around 0.1% of the time required for the same query on OLTP relational data. The most important mechanism in OLAP which allows it to achieve such performance is the use of *aggregations*. Aggregations are built from the fact table by changing the granularity on specific dimensions and aggregating up data along these dimensions, using an aggregate function (or *aggregation function*). The number of possible aggregations is determined by every possible combination of dimension granularities. The combination of all possible aggregations and the base data contains the answers to every query which can be answered from the data.

Because usually there are many aggregations that can be calculated, often only a predetermined number are fully calculated; the remainders are solved on demand. The problem of deciding which aggregations (views) to calculate is known as the view selection problem. View selection can be constrained by the total size of the selected set of aggregations, the time to update them from changes in the base data, or both. The objective of view selection is typically to minimize the average time to answer OLAP queries, although some studies also minimize the update time. View selection is NP-Complete. Many approaches to the problem have been explored, including greedy algorithms, randomized search, genetic algorithms and A* search algorithm.

Some aggregation functions can be computed for the entire OLAP cube by precomputing values fAPor each cell, and then computing the aggregation for a roll-up of cells by aggregating these aggregates, applying a divide and conquer algorithm to the multidimensional problem to compute them efficiently. For example, the overall sum of a roll-up is just the sum of the sub-sums in each cell. Functions that can be decomposed in this way are called decomposable aggregation functions, and include COUNT, MAX, MIN, and SUM, which can be computed for each cell and then directly aggregated; these are known as self-decomposable aggregation functions. In other cases the aggregate function can be computed by computing auxiliary numbers for cells, aggregating these auxiliary numbers, and finally computing the overall number at the end; examples include AVERAGE (tracking sum and count, dividing at the end) and RANGE (tracking max and min, subtracting at the end). In other cases the aggregate function cannot be computed without analyzing the entire set at once, though in some cases approximations can be computed; examples include DISTINCT COUNT, MEDIAN, and MODE; for example, the median of a set is not the median of medians of subsets. These latter are difficult to implement efficiently in OLAP, as they require computing the aggregate function on the base data, either computing them online (slow) or precomputing them for possible rollouts (large space).

Types

OL systems have been traditionally categorized using the following taxonomy.

Multidimensional OLAP (MOLAP)

Other MOLAP tools, particularly those that implement the functional database model do not pre-compute derived data but make all calculations on demand other than those that were previously requested and stored in a cache.

Advantages of MOLAP

- Fast query performance due to optimized storage, multidimensional indexing and caching.

- Smaller on-disk size of data compared to data stored in relational database due to compression techniques.
- Automated computation of higher level aggregates of the data.
- It is very compact for low dimension data sets.
- Array models provide natural indexing.
- Effective data extraction achieved through the pre-structuring of aggregated data.

Disadvantages of MOLAP

- Within some MOLAP systems the processing step (data load) can be quite lengthy, especially on large data volumes. This is usually remedied by doing only incremental processing, i.e., processing only the data which have changed (usually new data) instead of reprocessing the entire data set.
- Some MOLAP methodologies introduce data redundancy.

Relational OLAP (ROLAP)

ROLAP works directly with relational databases and does not require pre-computation. The base data and the dimension tables are stored as relational tables and new tables are created to hold the aggregated information. It depends on a specialized schema design. This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. In essence, each action of slicing and dicing is equivalent to adding a "WHERE" clause in the SQL statement. ROLAP tools do not use pre-calculated data cubes but instead pose the query to the standard relational database and its tables in order to bring back the data required to answer the question. ROLAP tools feature the ability to ask any question because the methodology is not limited to the contents of a cube. ROLAP also has the ability to drill down to the lowest level of detail in the database. While ROLAP uses a relational database source, generally the database must be carefully designed for ROLAP use. A database which was designed for OLTP will not function well as a ROLAP database. Therefore, ROLAP still involves creating an additional copy of the data. However, since it is a database, a variety of technologies can be used to populate the database.

Advantages of ROLAP

ROLAP is considered to be more scalable in handling large data volumes, especially models with dimensions with very high cardinality (i.e., millions of members).

- With a variety of data loading tools available, and the ability to fine-tune the extract, transform, load (ETL) code to the particular data model, load times are generally much shorter than with the automated MOLAP loads.
- The data are stored in a standard relational database and can be accessed by any SQL reporting tool (the tool does not have to be an OLAP tool).
- ROLAP tools are better at handling *non-aggregatable facts* (e.g., textual descriptions). MOLAP tools tend to suffer from slow performance when querying these elements.
- By decoupling the data storage from the multi-dimensional model, it is possible to successfully model data that would not otherwise fit into a strict dimensional model.
- The ROLAP approach can leverage database authorization controls such as row-level security, whereby the query results are filtered depending on preset criteria applied, for example, to a given user or group of users (SQL WHERE clause).

Disadvantages of ROLAP

- There is a consensus in the industry that ROLAP tools have slower performance than MOLAP tools. However, see the discussion below about ROLAP performance.
- The loading of *aggregate tables* must be managed by custom ETL code. The ROLAP tools do not help with this task. This means additional development time and more code to support.
- When the step of creating aggregate tables is skipped, the query performance then suffers because the larger detailed tables must be queried. This can be partially remedied by adding

additional aggregate tables; however it is still not practical to create aggregate tables for all combinations of dimensions/attributes.

- ROLAP relies on the general purpose database for querying and caching, and therefore several special techniques employed by MOLAP tools are not available (such as special hierarchical indexing). However, modern ROLAP tools take advantage of latest improvements in SQL language such as CUBE and ROLLUP operators, DB2 Cube Views, as well as other SQL OLAP extensions. These SQL improvements can mitigate the benefits of the MOLAP tools.
- Since ROLAP tools rely on SQL for all of the computations, they are not suitable when the model is heavy on calculations which don't translate well into SQL. Examples of such models include budgeting, allocations, financial reporting and other scenarios.

UNIT-III

Temporal Databases

A **temporal database** stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time. Temporal databases could be uni-temporal, bi-temporal or tri-temporal. More specifically the temporal aspects usually include valid time, transaction time or decision time

- **Valid time** is the time period during which a fact is true in the real world.
- **Transaction time** is the time period during which a fact stored in the database was known.
- **Decision time** is the time period during which a fact stored in the database was decided to be valid.

Packing and Unpacking Relations

Every element is a ***unit interval*** (i.e., consists of a single point

So the expanded form of {[1:2], [4:7], [6:9]} is {[1:1], [2:2], [4:4], [5:5], [6:6], [7:7], [8:8], [9:9]}.

Canonical forms for relations with one or more interval-valued attributes Based on collapsed and expanded forms. Both forms avoid redundancy.

Packed form of SD_PART “on DURING”:

SD_PART

S#	DURING
S2	[*d02*:*d04*]
S2	[*d03*:*d05*]
S4	[*d02*:*d05*]
S4	[*d04*:*d06*]
S4	[*d09*:*d10*]

Unpacked Form

S#	DURING
S2	[*d02*:*d02*]
S2	[*d03*:*d03*]
S2	[*d04*:*d04*]
S2	[*d05*:*d05*]
S4	[*d02*:*d02*]
S4	[*d03*:*d03*]
S4	[*d04*:*d04*]
S4	[*d05*:*d05*]
S4	[*d06*:*d06*]
S4	[*d09*:*d09*]
S4	[*d10*:*d10*]

Database Design

Conceptual schema

The physical design of the database specifies the physical configuration of the database on the storage media. This includes detailed specification of data elements, data types, indexing options and other parameters residing in the DBMS data dictionary. It is the detailed design of a system that includes modules & the database's hardware & software specifications of the system. Some aspects that are addressed at the physical layer:

- Security - end-user, as well as administrative security.
- Replication - what pieces of data get copied over into another database, and how often Are there multiple-masters, or a single one?
- High-availability - whether the configuration is active-passive, or active-active, the topology, coordination scheme, reliability targets, etc all have to be defined.
- Partitioning - if the database is distributed, then for a single entity, how is the data distributed amongst all the partitions of the database, and how is partition failure taken into account.
- Backup and restore schemes.

At the application level, other aspects of the physical design can include the need to define stored procedures, or materialized query views, OLAP cubes, etc.

Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint

1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Multimedia Databases

The multimedia databases are used to store multimedia data such as images, animation, audio, video along with text. This data is stored in the form of multiple file types like .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc.

Contents of the Multimedia Database

The multimedia database stored the multimedia data and information related to it. This is given in detail as follows:

Media data

This is the multimedia data that is stored in the database such as images, videos, audios, animation etc.

Media format data

The Media format data contains the formatting information related to the media data such as sampling rate, frame rate, encoding scheme etc.

Media keyword data

This contains the keyword data related to the media in the database. For an image the keyword data can be date and time of the image, description of the image etc.

Media feature data

The Media feature data describes the features of the media data. For an image, feature data can be colors of the image, textures in the image etc.

Challenges of Multimedia Database

There are many challenges to implement a multimedia database. Some of these are:

1. Multimedia databases contains data in a large type of formats such as .txt (text), .jpg (images), .swf (videos), .mp3(audio) etc. It is difficult to convert one type of data format to another.
2. The multimedia database requires a large size as the multimedia data is quite large and needs to be stored successfully in the database.
3. It takes a lot of time to process multimedia data so multimedia database is slow.

Multimedia Sources

The term *news* media refers to the groups that communicate information and news to people. Most Americans get their information about government from the news media because it would be impossible to gather all the news themselves. Media outlets have responded to the increasing reliance of Americans on television and the Internet by making the news even more readily available to people. There are three main types of news media: print media, broadcast media, and the Internet.

Print Media

The oldest media forms are newspapers, magazines, journals, newsletters, and other printed material. These publications are collectively known as the print media**.** Although print media readership has declined in the last few decades, many Americans still read a newspaper every day or a newsmagazine on a regular basis. The influence of print media is therefore significant. Regular readers of print media tend to be more likely to be politically The print media is responsible for more reporting than other news sources. Many news reports on television, for example, are merely follow-up stories about news that first appeared in newspapers. The top American newspapers, such as the *New York Times,* the *Washington Post,* and the *Los Angeles Times,* often set the agenda for many other media sources.

The Newspaper of Record

Because of its history of excellence and influence, the New York Times is sometimes called the newspaper of record: If a story is not in the Times, it is not important. In 2003, however, the newspaper suffered a major blow to its credibility when Times journalist Jayson Blair admitted that he had fabricated some of his stories. The Times has since made extensive efforts to prevent any similar scandals, but some readers have lost trust in the paper.

Broadcast Media

Broadcast media are news reports broadcast via radio and television. Television news is hugely important in the United States because more Americans get their news from television broadcasts than from any other source.

Multimedia Database Queries:

Query Processing Techniques

1. Content Based Retrieval

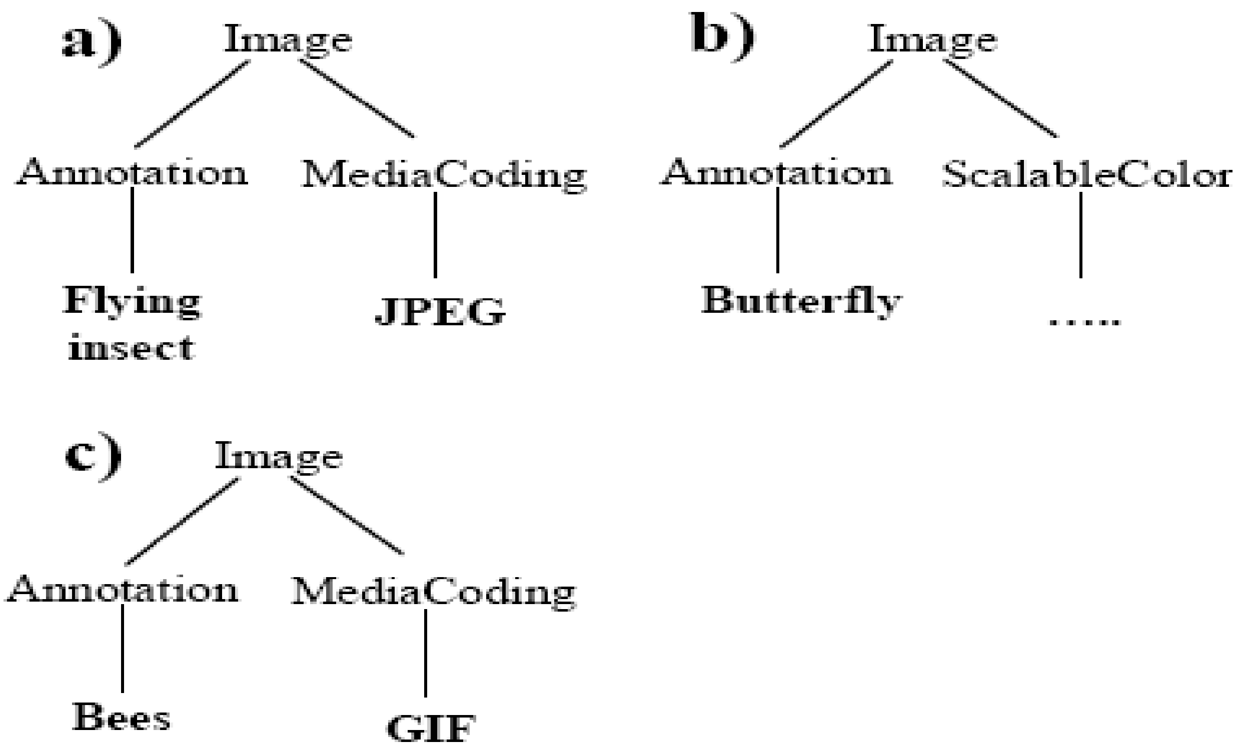
2. Tree Pattern Matching Retrieval

Query By Image Content (QBIC)

- QBIC was developed at IBM's Almaden Research Center.
- It is based on content based retrieval.
- IBM has already incorporated this technology into two of their products namely Ultimedia Manager and DB2 Extensions.

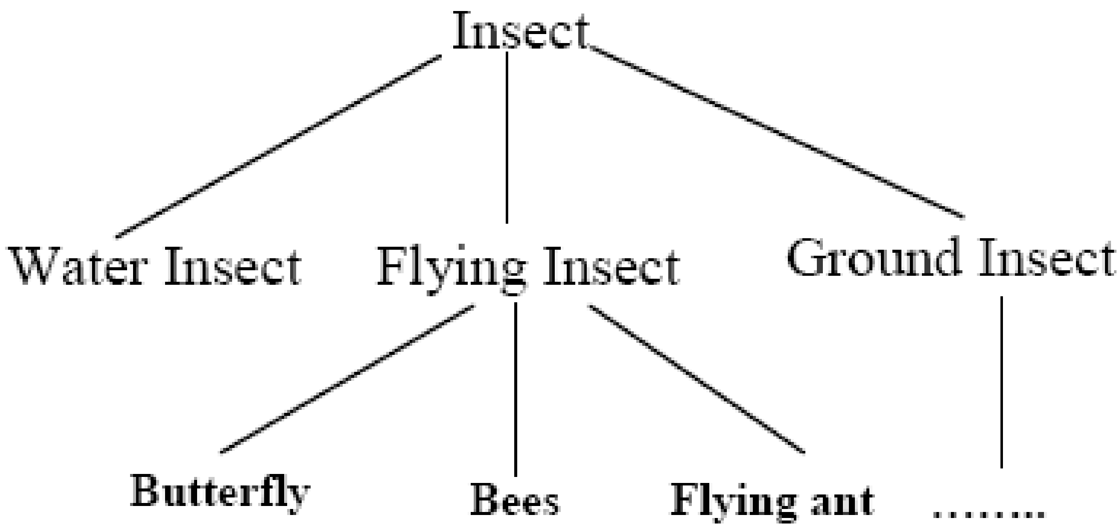
Tree Pattern Matching Rerieval

The major drawback of XML is that it cannot retrieve implicit data because XML does not have inference capabilities associated with its elements.



Tree Pattern Matching Approach

- Ontology is a data model that defines a set of classes and the relationships between those classes.



- **MPEG-7 Metadata Generator:** This component is used for the generation of metadata (color, size, etc) which is guided by the appropriate ontology.
- **MPEG-7 Query Generator:** This component is used to convert the user queries into MPEG-7 format.
- **Tree Generator:** This component is used to convert the MPEG-7 format query into a labeled ordered tree structure. A labeled tree is the one in which each node has specific label and an ordered tree is the one in which the parent child relationship and the left to right ordering among siblings are significant.

Searching Strategy: This component is based on the tree embedded approximation algorithm which is used to match the user query tree with the MPEG-7 data tree and retrieve the appropriate results for the user query.

Multimedia Database Applications

A digital library, digital repository, or digital collection, is an online database of digital objects that can include text, still images, audio, video, or other digital media formats. Objects can consist of digitized content like print or photographs, as well as originally produced digital content like word processor files or social media posts. In addition to storing content, digital libraries provide means for organizing, searching, and retrieving the content contained in the collection. Digital libraries can vary immensely in size and scope, and can be maintained by individuals or organizations. The digital content may be stored locally, or accessed remotely via computer networks. These information retrieval systems are able to exchange information with each other through interoperability and sustainability.

Video on-demand (**VOD**) is a video media distribution system that allows users to access video entertainment without a traditional video entertainment device and without the constraints of a typical static broadcasting schedule. In the 20th century, broadcasting in the form of over-the-air programming was the commonest form of media distribution. As Internet and IPTV technologies continued to develop in the 1990s, consumers began to gravitate towards non-traditional modes of content consumption, which culminated in the arrival of VOD on televisions and personal computers. Television VOD systems can stream content, either through a traditional set-top box or through remote devices such as computers, tablets, and smartphones. VOD users can permanently download content to a device such as a computer, digital video recorder or a portable media player for continued viewing. The majority of cable and telephone company-based television providers offer VOD streaming, whereby a user selects a video program that begins to play immediately or downloading to a digital video recorder (DVR) rented or purchased from the provider, or to a PC or to a portable device for delayed viewing.

Internet television has emerged as an increasingly popular medium of VOD provision. Desktop client applications such as the Apple iCloud online content store and Smart TV apps such as Amazon Prime Video allow temporary rentals and purchases of video entertainment content. Other internet-based VOD systems provide users with access to bundles of video entertainment content rather than individual movies and shows. The most common of these systems, Netflix and Hulu, use a subscription model that requires users to pay a monthly fee for access to bundle of movies, television shows, and original series. In contrast, YouTube, another internet-based VOD system, uses an advertising-funded model in which users can access most of YouTube's video content free of charge but must pay a subscription fee for premium content.

Some airlines offer VOD services as in-flight entertainment to passengers through video screens embedded in seats or externally provided portable media players.

UNIT-IV

Spatial Database

Spatial Data:

Spatial data, also known as geospatial data, is information about a physical object that can be represented by numerical values in a geographic coordinate system. Spatial data represents the location, size and shape of an object on planet Earth such as a building, lake, mountain or township

Spatial Database Characteristics

Database systems use indexes to quickly look up values; however, this way of indexing data is not optimal for spatial queries. Instead, spatial databases use a spatial index to speed up database operations. In addition to typical SQL queries such as SELECT statements, spatial databases can perform a wide variety of spatial operations. The following operations and many more are specified by the Open Geospatial Consortium standard:

- Spatial Measurements: Computes line length, polygon area, the distance between geometries, etc.
- Spatial Functions: Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.
- Spatial Predicates: Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a residence located within a mile of the area we are planning to build the landfill?'
- Geometry Constructors: Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.

- Observer Functions: Queries which return specific information about a feature such as the location of the center of a circle

Some databases support only simplified or modified sets of these operations, especially in cases of NoSQL systems like MongoDB and CouchDB.

Spatial index

Spatial indices are used by spatial databases (databases which store information related to objects in space) to optimize spatial queries. Conventional index types do not efficiently handle spatial queries such as how far two points differ, or whether points fall within a spatial area of interest. Common spatial index methods include:

- Geohash
- HHCode
- Grid (spatial index)
- Z-order (curve)
- Quadtree
- Octree
- UB-tree
- R-tree: Typically the preferred method for indexing spatial data Objects (shapes, lines and points) are grouped using the minimum bounding rectangle (MBR). Objects are added to an MBR within the index that will lead to the smallest increase in its size.
- R+ tree
- R* tree

Spatial Data Model

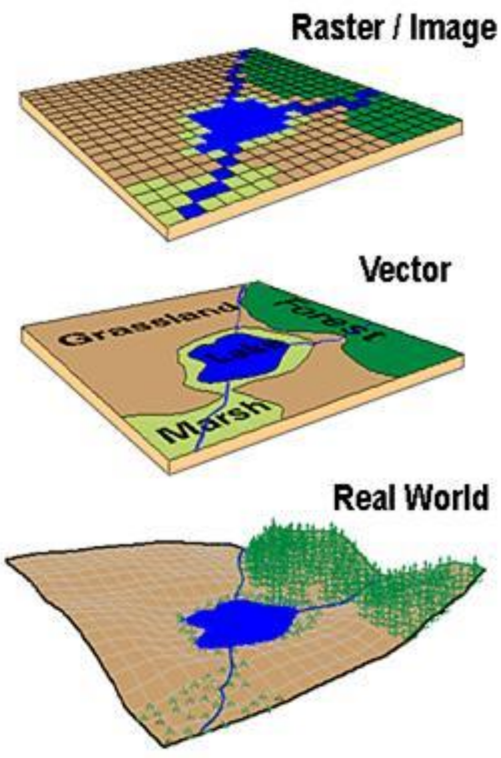
A data model is a way of defining and representing real world surfaces and characteristics in GIS. There are two primary types of spatial data models: Vector and Raster.

SPATIAL DATA MODELS

Traditionally spatial data has been stored and presented in the form of a map. Three basic types of spatial data models have evolved for storing geographic data digitally. These are referred to as:

1. Vector
2. Raster and
3. Image.

The following diagram reflects the two primary spatial data encoding techniques. These are vector and raster. Image data utilizes techniques very similar to raster data, however typically lacks the internal formats required for analysis and modeling of the data. Images reflect *pictures* or *photographs* of the landscape.



Spatial Database Queries

A spatial query is a special type of database query supported by geodatabases and spatial databases. The queries differ from non-spatial SQL queries in several important ways.

Spatial query

A spatial query is a special type of database query supported by geodatabases and spatial databases. The queries differ from non-spatial SQL queries in several important ways. Two of the most important are that they allow for the use of geometry data types such as points, lines and polygons and that these queries consider the spatial relationship between these geometries.

Types of queries

The function names for queries differ across geodatabases. The following list contains commonly used functions built into PostGIS, a free geodatabase which is a PostgreSQL extension (the term 'geometry' refers to a point, line, box or other two or three dimensional shape):

Function prototype: *functionName (parameter(s)) : return type*

- Distance(geometry, geometry) : number
- Equals(geometry, geometry) : boolean
- Disjoint(geometry, geometry) : boolean
- Intersects(geometry, geometry) : boolean
- Touches(geometry, geometry) : boolean
- Crosses(geometry, geometry) : boolean
- Overlaps(geometry, geometry) : boolean
- Contains(geometry, geometry) : boolean
- Length(geometry) : number
- Area(geometry) : number
- Centroid(geometry) : geometry

Logic based Databases:

A deductive database is a database system that can make deductions (i.e., conclude additional facts) based on rules and facts stored in the (deductive) database. Datalog is the language typically used to specify facts, rules and queries in deductive databases. Deductive databases have grown out of the desire to combine logic programming with relational databases to construct systems that support a powerful formalism and are still fast and able to deal with very large datasets. Deductive databases are more expressive than relational databases but less expressive than logic programming systems. In recent years, deductive databases such as Datalog have found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

Deductive databases and logic programming: Deductive databases reuse many concepts from logic programming; rules and facts specified in the deductive database language Datalog look very similar to those in Prolog. However important differences between deductive databases and logic programming:

- Order sensitivity and procedurality: In Prolog, program execution depends on the order of rules in the program and on the order of parts of rules; these properties are used by programmers to build efficient programs. In database languages (like SQL or Datalog), however, program execution is independent of the order of rules and facts.
- Special predicates: In Prolog, programmers can directly influence the procedural evaluation of the program with special predicates such as the cut, this has no correspondence in deductive databases.
- Function symbols: Logic Programming languages allow function symbols to build up complex symbols. This is not allowed in deductive databases.
- Tuple-oriented processing: Deductive databases use set-oriented processing while logic programming languages concentrate on one tuple at a time.

Propositional Calculus:

Propositional calculus is a branch of logic. It is also called propositional logic, **,** statement logic sentential calculus, sentential logic, or sometimes zeroth-order logic. It deals with propositions (which can be true or false) and argument flow. Compound propositions are formed by connecting propositions by logical connectives. The propositions without logical connectives are called atomic propositions. Unlike first-order logic, propositional logic does not deal with non-logical objects, predicates about them, or quantifiers. However, all the machinery of propositional logic is included in first-order logic and higher-order logics. In this sense, propositional logic is the foundation of first-order logic and higher-order logic.

Predicate Calculus

First-order logic—also known as predicate logic, quantificational logic, and first-order predicate calculus—is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science. First-order logic uses quantified variables over non-logical objects and allows the use of sentences that contain variables, so that rather than propositions such as *Socrates is a man* one can have expressions in the form "there exists x such that x is Socrates and x is a man" and *there exists* is a quantifier while *x* is a variable. This distinguishes it from propositional logic, which does not use quantifiers or relations In this sense, propositional logic is the foundation of first-order logic. A theory about a topic is usually a first-order logic together with a specified domain of discourse over which the quantified variables range, finitely many functions from that domain to itself, finitely many predicates defined on that domain, and a set of axioms believed to hold for those things. Sometimes "theory" is understood in a more formal sense, which is just a set of sentences in first-order logic. The adjective "first-order" distinguishes first-order logic from higher-order logic in which there are predicates having predicates or functions as arguments, or in which one or both of predicate quantifiers or function quantifiers are permitted. In first-order theories, predicates are often associated with sets. In interpreted higher-order theories, predicates may be interpreted as sets of sets.

There are many deductive systems for first-order logic which are both sound (all provable statements are true in all models) and complete (all statements which are true in all models are provable). Although the logical consequence relation is only semidecidable, much progress has been made in automated theorem proving in first-order logic. First-order logic also satisfies several metalogical theorems that make it amenable to analysis in proof theory, such as the Löwenheim–Skolem theorem and the compactness theorem. First-order logic is the standard for the formalization of mathematics into axioms and is studied in the foundations of mathematics. Peano arithmetic and Zermelo–Fraenkel set theory are axiomatizations of number theory and set theory, respectively, into first-order logic. No first-order theory, however, has the strength to uniquely describe a structure with an infinite domain, such as the natural numbers or the real line. Axiom systems that do fully describe these two structures (that is, categorical axiom systems) can be obtained in stronger logics such as second-order logic.

Recursive Query Processing

Recursive query processing methods can be broadly categorized as the bottom-up methods, or top-down methods. Bottom-up methods answer a query by applying all rules of a program to ground tuples, deriving tuples that satisfy rule bodies into predicates in rule heads. The minimal model for the given program and ground tuples is explicitly materialized as a new database instance; the answer to the query is then obtained through a simple select/project/join operations over the materialized database instance. In contrast, top-down methods answer a query by pushing selection criteria (i.e. constants) from the query down into rules that may answer the query (i.e. rules deriving into predicates being queried), creating more (sub)queries from the atoms of these rules' bodies; the subqueries are in turn answered in a similar, top-down fashion.

UNIT-V

Internet Database

An Internet database is a database accessible from a local network or the Internet, as opposed to one that is stored locally on an individual computer or its attached storage (such as a CD). Online databases are hosted on websites, made available as software as a service products accessible via a web browser. They may be free or require payment, such as by a monthly subscription. Some have enhanced features such as collaborative editing and email notification.

Multimedia Database

A Multimedia database (MMDB) is a collection of related for multimedia data. The multimedia data include one or more primary media data types such as text, images, graphic objects (including drawings, sketches and illustrations) animation sequences, audio and video. A Multimedia Database Management System (MMDBMS) is a framework that manages different types of data potentially represented in a wide diversity of formats on a wide array of media sources. It provides support for multimedia data types, and facilitate for creation, storage, access, query and control of a multimedia database.

Requirements of Multimedia databases

Like the traditional databases, Multimedia databases should address the following requirements:

- Integration
    - Data items do not need to be duplicated for different programs invocations
- Data independence
    - Separate the database and the management from the application programs
- Concurrency control
    - Allows concurrent transactions
- Persistence

    - Data objects can be saved and re-used by different transactions and program invocations
- Privacy
    - Access and authorization control
- Integrity control
    - Ensures database consistency between transactions
- Recovery
    - Failures of transactions should not affect the persistent data storage
- Query support
    - Allows easy querying of multimedia data

Multimedia databases should have the ability to uniformly query data (media data, textual data) represented in different formats and have the ability to simultaneously query different media sources and conduct classical database operations across them. They should have the ability to retrieve media objects from a local storage device in a good manner. They should have the ability to take the response generated by a query and develop a presentation of that response in terms of audio-visual media and have the ability to deliver this presentation.

Mobile Database

Mobile computing devices (e.g., smartphones and PDAs) store and share data over a mobile network, or a database which is actually stored by the mobile device. This could be a list of contacts, price information, distance travelled, or any other information. Many applications require the ability to download information from an information repository and operate on this information even when out of range or disconnected. An example of this is your contacts and calendar on the phone. In this scenario, a user would require access to update information from files in the home directories on a server or customer records from a database. This type of access and work load generated by such users is different from the traditional workloads seen in client–server systems of today. Mobile databases are not used solely for the revision of company contacts and calendars, but used in a number of industries.

Mobile Database is a database that is transportable, portable and physically separate or detached from the corporate database server but has the capability to communicate with those servers from remote sites allowing the sharing of various kinds of data. With mobile databases, users have access to corporate data on their laptop, PDA, or other Internet access device that is required for applications at remote sites.

The components of a mobile database environment include:

- Corporate database server and DBMS that deals with and stores the corporate data and provides corporate applications
- Remote database and DBMS usually manages and stores the mobile data and provides mobile applications
- mobile database platform that includes a laptop, PDA, or other Internet access devices
- Two-way communication links between the corporate and mobile DBMS.

Based on the particular necessities of mobile applications, in many of the cases, the user might use a mobile device may and log on to any corporate database server and work with data there, while in others the user may download data and work with it on a mobile device or upload data captured at the remote site to the corporate database. The communication between the corporate and mobile databases is usually discontinuous and is typically established or gets its connection for a short duration of time at irregular intervals. Although unusual, some applications require direct communication between the mobile databases. The two main issues associated with mobile databases are the management of the mobile database and the communication between the mobile and corporate databases. In the following section, we identify the requirements of mobile DBMSs. The additional functionality required for mobile DBMSs includes the capability to:

- communicate with the centralized or primary database server through modes
- repeat those data on the centralized database server and mobile device
- coordinate data on the centralized database server and mobile device
- capture data from a range of sources such as the Internet

- deal with those data on the mobile device
- analyze those data on a mobile device
- create customized and personalized mobile applications

MY SQL

MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications. MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.

- MySQL is an open-source database so you don't have to pay a single penny to use it.
- MySQL is a very powerful program so it can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MYSQL Database

MySQL Create Database

A database is a collection of data. MySQL allows us to store and retrieve the data from the database in a efficient way. In MySQL, we can create a database using the CREATE DATABASE statement. But, if database already exits, it throws an error. To avoid the error, we can use the IF NOT EXISTS option with the CREATE DATABASE statement.

Syntax:

1. CREATE DATABASE database_name;

Example:

Let's take an example to create a database name "employees"

1. CREATE DATABASE employees;

MySQL SELECT Database

SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command USE to select a particular database.

Syntax:

1. USE database_name;

Example:

Let's take an example to use a database name "customers".

1. USE customers;

MySQL Drop Database

We can drop/delete/remove a MySQL database easily with the MySQL DROP DATABASE command. It deletes all the tables of the database along with the database permanently. It throws an error, if the database is not available. We can use the IF EXISTS option with the DROP DATABASE statement. It returns the numbers of tables which are deleted through the DROP DATABASE statement. We should be careful while deleting any database because we will loose all the data available in the database.

Syntax:

1. DROP DATABASE database_name;

Example:

Let's take an example to drop a database name "employees"

1. DROP DATABASE employees;