UNIT III ADVANCED DATABASE SYSTEMS

SYLLABUS

Object Oriented Databases-Need for Complex Data Types - The Object-Oriented Data Model-Object-Oriented Languages-Spatial Databases: Spatial Data Types, Spatial Relationships, Spatial Data Structures, Spatial Access Methods – Temporal Databases: Overview – Active Databases – Deductive Databases – Recursive Queries in SQL – Mobile Databases: Location and Handoff Management, Mobile Transaction Models, Concurrency – Transaction Commit Protocols – Multimedia Databases.

SNAPSHOT:

1. Object-Oriented Databases
    i. Need for Complex Data Types
    ii. The Object-Oriented Data Model
    iii. Object-Oriented Languages
2. Spatial Databases
    i. Spatial Data Types
    ii. Spatial Relationships
    iii. Spatial Data Structures
    iv. Spatial Access Methods
3. Temporal Databases
    i. Overview
4. Active Databases
5. Deductive Databases
6. Recursive Queries in SQL
7. Mobile Databases
    i. Location and Handoff Management
    ii. Mobile Transaction Models
    iii. Concurrency
8. Transaction Commit Protocols
9. Multimedia Databases

CONTEXT

OBJECT ORIENTED DATABASES

Object-oriented databases have adopted many of the concepts that were developed Originally for object-oriented programming languages.

The key concepts utilized in many object database systems are detailed below,

CONCEPTS

OBJECT IDENTITY

- an ODMS provides a unique identity to each independent object stored in the data-Base.
- This unique identity is typically implemented via a unique, system-generated Object identifier (OID).
- The value of an OID is not visible to the external user, but is used internally by the system to identify each object uniquely and to create and manage inter-object references.
- The main property required of an OID is that it be immutable; that is, the OID Value of a particular object should not change.

COMPLEX TYPE STRUCTURES

- In ODBs, a complex type may be constructed from other types by nesting of type constructors.
- The three most basic constructors are Atom, struct (or tuple), and collection

ENCAPSULATION

- It is also related to the concepts of Abstract data types and information hiding in programming languages.
- The object attribute and methods are encapsulated together.
- Methods defines the behaviour of a Type of object based on the operations that can be externally applied to objects
- Operations May perform a retrieval, calculation, and update or combination of all.
- The external users of the object are only made aware of the interface of the operations, which defines the name and arguments (parameters) of each operation. The Implementation is hidden from the external users.

OBJECT PERSISTENCE

- Persistent objects are stored in the database and persist after Program termination. The typical mechanisms for making an object persistent are Naming and reachability.
- The naming mechanism involves giving an object a unique persistent name within A particular database.

- The named persistent Objects are used as entry points to the database through which users and applications can start their database access.
- Obviously, it is not practical to give names to all
  Objects in a large database that includes thousands of objects, so most objects are
  Made persistent by using the second mechanism, called reachability.
- The reachability mechanism works by making the object reachable from some other persistent Object. An object B is said to be reachable from an object A if a sequence of references in the database lead from object A to object B.

INHERITANCE.

- Inheritance allows the Definition of new types based on other predefined types, leading to a type (or class) Hierarchy.
- A type is defined by assigning it a type name, and then defining several attributes (instance variables) and operations (methods) for the type.

PERSON: Name, Address, Birthdate, Age, Ssn

In the PERSON type, the Name, Address, Ssn, and Birthdate functions can be implemented as stored attributes, whereas the Age function can be implemented as an operation that calculates the Age from the value of the Birthdate attribute and the Current date.

EMPLOYEE: Name, Address, Birthdate, Age, Ssn, Salary, Hire date, Seniority

STUDENT: Name, Address, Birthdate, Age, Ssn, Major, GPA

Since both STUDENT and EMPLOYEE include all the functions defined for PERSON Plus some additional functions of their own, we can declare them to be subtypes of PERSON. Each will inherit the previously defined functions of PERSON—namely, Name, Address, Birthdate, Age, and Ssn.

Therefore, we can declare EMPLOYEE and STUDENT as follows:

EMPLOYEE subtype-of PERSON: Salary, Hire date, Seniority

STUDENT subtype-of PERSON: Major, GPA

EXTENTS CORRESPONDING TO A TYPE HIERARCHY

- An extent is defined to store the collection of persistent objects for each type or subType
- An extent is a named persistent object whose value is a persistent collection that Holds a collection of objects of the same type that are stored permanently in the Database
- Denoted by the keyword EXTEND.

POLYMORPHISM

- Is also known as operator overloading.
- This concept allows the same Operator name or symbol to be bound to two or more different implementations of the operator, depending on the type of objects to which the operator is applied.
- In some Languages, the operator symbol "+" can mean different things when applied to
    - Operands (objects) of different types. If the operands of "+" are of type integer, the operation invoked is integer addition.
    - If the operands of "+" are of type floating Point, the operation invoked is floating point addition.
    - If the operands of "+" are of Type set, the operation invoked is set union.

MULTIPLE INHERITANCE

- Multiple inheritance occurs When a certain subtype T is a subtype of two (or more) types and hence inherits the Functions (attributes and methods) of both supertypes.
- For example, we may create a Subtype ENGINEERING_MANAGER that is a subtype of both MANAGER and ENGINEER.

COMPLEX DATA TYPES

A complex data type *is usually a composite of other existing data types.*

An important advantage that complexes data types have over user-defined types is that users can access and manipulate the individual components of a complex data type.

In contrast, built-in types and user-defined types are self-contained (encapsulated) data types. Consequently, the only way to access the component values of an opaque data type is through functions that you define on the opaque type.

Complex types are supported by most programming languages including Python, C++, and Java. They are also supported by databases

Examples of Complex Types

1. **Array**
    - Array is a homogenous collection of elements of the same data type.

- These can be int, float, or any kind of string (string, char, varchar, etc.) or another complex type.
- The Array data type is quite like Java's array list which is a strongly typed list but also lets you nest the lists to create arrays.

2. **Map**
    - Map data types are key-value pairs.
    - This type associates a key with a value.
    - For example, a city could be associated with a zip code, or an address with a phone number.
    - The keys can only be made of the base types (not a complex type). Map values can be any type that Array supports. For example, the key could be a string and the value could be an int.
3. **Struct**
    - Struct is a composite data type that can contain other complex and simple data types.
    - The data types in a struct do not need to be the same (as they do in arrays.)
    - This flexibility allows users to combine multiple data types together under a single name. While many other member variables of similar or different data types can be part of a struct, all these variables need to be defined at the time the struct is created.
4. **Nested Complex Types**
    - Can also create complex types by nesting other complex types.

NEED FOR COMPLEX DATA TYPES

Complex data types in object-oriented databases are essential for representing and managing complex and structured data in a flexible and efficient manner. They provide a way to model real-world entities and their relationships more accurately and comprehensively. Here are some reasons for the need for complex data types in object-oriented databases:

1. **Enhanced Data Modelling**: Complex data types allow for more expressive data modelling compared to simple data types like integers and strings. They enable the representation of complex relationships and structures, such as hierarchies, networks, graphs, and recursive relationships. This allows for a more accurate representation of real-world entities and their relationships.
2. **Encapsulation and Data Integrity:** Complex data types facilitate encapsulation by bundling related attributes and behaviours into a single object. This ensures data integrity and consistency by enforcing constraints and rules at the object level. For example, an object representing a bank account can encapsulate its attributes (e.g., balance, account number) and behaviours (e.g., deposit, withdrawal) together, ensuring that the data and operations related to an account are well-defined and controlled.

3. **Object Reusability:** Complex data types enable the creation of reusable components and object hierarchies. By defining complex data types, such as classes or structures, with their associated attributes and behaviours, they can be instantiated and reused throughout the database. This promotes code reuse, reduces redundancy, and improves the overall maintainability of the system.
4. **Data Abstraction and Polymorphism:** Complex data types support data abstraction, allowing the hiding of implementation details and exposing only the necessary interfaces. This promotes modularity and encapsulation, enabling different parts of the system to interact with complex data types through well-defined interfaces. Polymorphism, a fundamental principle of object-oriented programming, allows objects of different complex types to be treated interchangeably, enhancing flexibility and extensibility.
5. **Querying and Data Manipulation:** Complex data types provide more expressive querying capabilities. Object-oriented databases typically offer query languages that allow for querying and manipulating complex data structures directly. For example, a query language might support traversing object hierarchies, filtering objects based on complex criteria, and performing complex operations like joins and aggregations.
6. **Performance Optimization:** Complex data types can be optimized for specific operations and access patterns. For example, a database can provide specialized indexing mechanisms for efficient searching and retrieval of complex data structures. Additionally, object-oriented databases can leverage object-level caching and memory management techniques to optimize performance when working with complex objects.

Overall, complex data types in object-oriented databases enable a more natural and comprehensive representation of data, support encapsulation and modularity, enhance code reusability, and provide expressive querying and manipulation capabilities. They form the foundation for building sophisticated and flexible data models that align closely with the complexities of real-world domains.

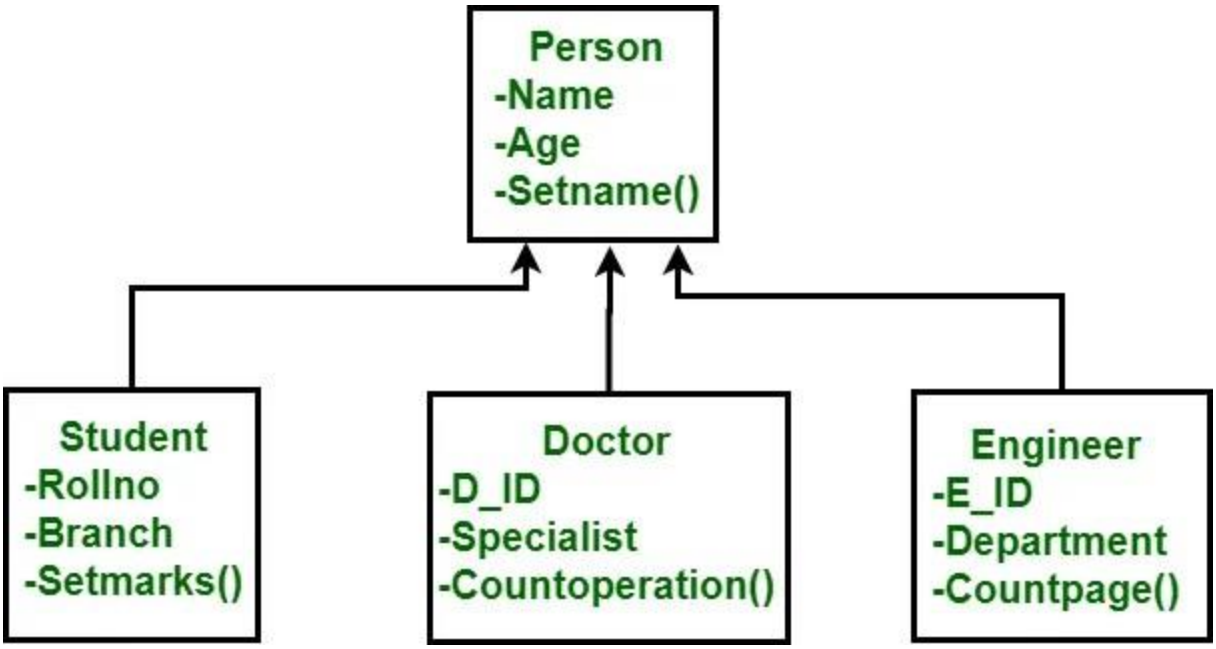
THE OBJECT-ORIENTED DATA MODEL

To represent the complex real-world problems there was a need for a data model that is closely related to real world. Object Oriented Data Model represents the real-world problems easily.

In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object

In this, real world problems are represented as objects with different attributes.

All objects have multiple relationships between them. Basically, it is combination of Object-Oriented programming and Relational Database Model.



COMPONENTS OF OBJECT-ORIENTED DATA MODEL

1. **Objects**
   - It is an instance of class.
   - An object is an abstraction of a real-world entity
   - Objects encapsulates data and code into a single unit which provide data abstraction by hiding the implementation details from the user.
   - For example: Instances of student, doctor, engineer in above figure.
2. **Attribute**
   - An attribute describes the properties of object.
   - For example: Object is STUDENT and its attribute are Roll no, Branch.
3. **Methods**
   - Method represents the behaviour of an object. Basically, it represents the real-world action.
   - For example: Finding a STUDENT marks in above figure
4. **Class**
   - A class is a collection of similar objects with shared structure i.e., attributes and behaviour i.e., methods.
   - An object is an instance of class. For example: Person, Student, Doctor, Engineer
5. **Inheritance**
   - By using inheritance, new class can inherit the attributes and methods of the old class i.e., base class.
   - For example: as classes Student, Doctor and Engineer are inherited from the base class Person.

ADVANTAGES AND DISADVANTAGES OF OBJECT-ORIENTED DATA MODEL:

The object-oriented data model has several advantages and disadvantages. Let us explore them:

Advantages

1. Encapsulation: The object-oriented data model promotes encapsulation by bundling related data and behaviour into objects. This encapsulation ensures data integrity, as objects are responsible for maintaining the consistency and validity of their internal state. It also provides a natural way to organize and manage complex data structures.

2. Reusability: Objects can be defined as reusable components that can be instantiated and used in various parts of an application or database. This promotes code reuse, reduces redundancy, and improves the overall maintainability of the system. Inheritance allows for the creation of hierarchies of objects, where subclasses inherit and extend the properties and behaviours of their parent classes, further enhancing reusability.

3. Modularity: The object-oriented data model encourages modular design by breaking down the system into smaller, self-contained objects. Each object encapsulates its data and behaviour, allowing for independent development, testing, and maintenance. This modular approach simplifies the complexity of large systems and promotes better organization and manageability of the codebase.

4. Flexibility and Extensibility: The object-oriented data model provides flexibility and extensibility by allowing the addition of new classes and objects without modifying existing ones. This supports a modular and evolutionary development process, where new features and functionalities can be added by creating new objects or subclasses without impacting the existing system.

5. Data Abstraction: Object-oriented databases support data abstraction, which means they can hide the implementation details and expose only the necessary interfaces to interact with objects. This abstraction simplifies the complexity of the underlying data structure, making it easier to understand and work with the data. It also allows for the separation of concerns between different parts of the system.

Disadvantages:

1. Performance Overhead: Object-oriented databases can introduce performance overhead compared to more efficient data models like relational databases. The overhead is mainly due to the need for object traversal and complex queries involving multiple objects. Additionally, the flexibility and abstraction provided by the object-oriented model may require additional processing and memory resources.

2. Complexity: The object-oriented data model can be more complex to understand and implement compared to simpler data models. It requires a good understanding of object-

oriented programming concepts, such as inheritance, polymorphism, and object relationships. The complexity can increase when dealing with large and interconnected object hierarchies.

3. Lack of Standardization: Unlike the widely adopted relational data model with SQL as a standard query language, object-oriented databases lack a standardized query language. Different object-oriented databases may have their own proprietary query languages, making it harder to switch between databases or integrate systems using different object-oriented databases.

4. Scalability and Interoperability: Scaling object-oriented databases to handle large amounts of data and high concurrency can be challenging. Distributed object-oriented databases face difficulties in maintaining consistency across distributed objects. Interoperability can also be an issue when integrating object-oriented databases with other systems or databases that follow different data models.

5. Learning Curve: The object-oriented data model requires a shift in mindset and a different approach to data modelling compared to traditional relational databases. Developers and database administrators may need to learn new concepts and techniques associated with object-oriented programming and design, which can be a steep learning curve for those accustomed to relational databases.

It is important to consider these advantages and disadvantages while evaluating the suitability of the object-oriented data model for a specific application or use case.

Merits

- Codes can be reused due to inheritance.
- Easily understandable.
- Cost of maintenance can reduce due to reusability of attributes and functions because of inheritance.

Demerits

- It is not properly developed so not accepted by users easily.

OBJECT-ORIENTED LANGUAGES

Object-oriented databases (OODBs) are designed to store, manage, and retrieve data using object-oriented concepts. These databases are closely aligned with object-oriented programming languages. Here are some popular object-oriented programming languages that are commonly used in conjunction with object-oriented databases:

1. **Java**: Java is a widely used programming language that aligns well with object-oriented databases. It provides features such as classes, objects, inheritance, and encapsulation, which are fundamental to the object-oriented data model. Java also offers robust support for connecting to

databases through technologies like JDBC (Java Database Connectivity) or ORM (Object-Relational Mapping) frameworks like Hibernate.

2. **C++:** C++ is a powerful language that supports object-oriented programming and is often used in conjunction with object-oriented databases. C++ allows the definition of classes, objects, and inheritance, providing a strong foundation for working with object-oriented data models. C++ also offers libraries and frameworks for database connectivity, such as ODBC (Open Database Connectivity) or C++ ORM frameworks like ODB or Qt SQL.
3. **C#:** C# (pronounced C-sharp) is a language developed by Microsoft specifically for the .NET framework. C# is designed to support object-oriented programming paradigms and provides features like classes, objects, inheritance, and encapsulation. It offers seamless integration with databases through ADO.NET (ActiveX Data Objects for .NET) or ORM frameworks like Entity Framework.
4. **Python**: Python is a versatile language that supports both procedural and object-oriented programming styles. Python's object-oriented features, such as classes, objects, inheritance, and polymorphism, make it well-suited for working with object-oriented databases. Python offers database connectivity through libraries like PyODBC, SQLAlchemy, or Django ORM.
5. **Ruby**: Ruby is a dynamic, reflective, and object-oriented language known for its simplicity and readability. It aligns well with object-oriented databases and provides features like classes, objects, inheritance, and mixins. Ruby supports database connectivity through libraries like ActiveRecord or DataMapper.
6. **Smalltalk**: Smalltalk is one of the earliest and purest object-oriented programming languages. It is closely aligned with the concepts of object-oriented databases and provides a seamless integration between programming and database models. Smalltalk allows for the direct manipulation and storage of objects in the database, making it an ideal language for working with object-oriented databases.

These are just a few examples of object-oriented programming languages that are commonly used in conjunction with object-oriented databases. Each language has its own strengths, features, and frameworks for connecting to databases. The choice of language depends on factors such as the requirements of the application, the specific object-oriented database being used, and the preferences and skills of the development team.

SPATIAL DATABASES

- Spatial databases incorporate functionality that provides support for databases that Keep track of objects in a multidimensional space.
- In general, a Spatial Database stores objects that have spatial characteristics that describe them and that Have spatial relationships among them.
- The spatial relationships among the objects Are important, and they are often needed when querying the database.
- A Spatial database can in general refer to an n-dimensional space.

For example,

- Satellite images are a prominent example of Spatial data
- cartographic data-Bases that store maps include two-dimensional spatial descriptions of their Objects—from countries and states to rivers, cities, roads, seas, and so on.

SPATIAL QUERIES

- Queries posed on these spatial data, where predicates for selection deal with spatial parameters, are called spatial queries.
- For example, "What are the Names of all bookstores within five miles of the College of Computing building at Georgia Tech?" is a spatial query.

There is a special need for databases tailored for handling Spatial data and spatial queries.

- Measurement operations are used to measure some global properties of single objects (such as the area, the relative size of an object's Parts, compactness, or symmetry), and to measure the relative position of different Objects in terms of distance and direction.
- Spatial analysis operations, which often Use statistical techniques, are used to uncover spatial relationships within and among mapped data layers.
  An example would be to create a map—known as a Prediction map—that identifies the locations of likely customers for Products based on the historical sales and demographic information.
- Flow analysis Operations help in determining the shortest path between two points and the Connectivity among nodes or regions in a graph.
- Location analysis aims to find if the given set of points and lines lie within a given polygon (location).
- Spatial search allows a user to search for objects within a particular Spatial region.
- Digital terrain analysis is used to build three-dimensional models, Where the topography of a geographical location can be represented with an x, y, z. Data model known as Digital Terrain (or Elevation) Model (DTM/DEM). The x and Y dimensions of a DTM represent the horizontal plane, and z represents spot Heights for the respective x, y coordinates.

SPATIAL DATA TYPES

Spatial data comes in three basic forms.

1. **Map Data**

- Map Data includes various geographic or spatial features of objects in a Map, such as an object's shape and the location of the object within the map.
- The three basic types of features are
  i) **Points**
    (1) Points are used to represent spatial characteristics of objects whose locations Correspond to a single 2-d coordinate (x, y, or longitude/latitude) in the scale of a particular application.
    (2) Some examples of stationary point Objects could be buildings, cellular towers, or stationary vehicles,
    (3) Moving Vehicles and other moving objects can be represented by a sequence of point Locations that change over time.
  ii) **Lines**

(1) Lines represent objects having length, such as roads or rivers, whose spatial characteristics can be approximated by a Sequence of connected lines

iii) Polygon

(1) Polygons are used to represent spatial characteristics of objects that have a boundary, such as countries, states, lakes, or Cities. Notice that some objects, such as buildings or cities, can be represented as either points or polygons, depending on the scale of details.

2. **Attribute data**

- Attribute data is the descriptive data that GIS systems associate with map Features.
- For example, suppose that a map contains features that represent Counties within a US state (such as Texas or Oregon).
- Attributes for each County feature (object) could include population, largest city/town, area in square miles, and so on.

3. Image data

- Image data includes data such as satellite images and aerial photographs, which are typically created by cameras.
- Objects of interest, such as buildings
  And roads, can be identified and overlaid on these images.
- Images can also be Attributes of map features. One can add images to other map features so that Clicking on the feature would display the image. Aerial and satellite images Are typical examples of raster data.

SPATIAL RELATIONSHIPS,

Spatial operators are used to capture all the relevant geometric properties of objects Embedded in the physical space and the relations between them, as well as to Perform spatial analysis.

1. **Topological operators**.

- Topological operators are Hierarchically structured in several levels, where the base level offers operators the ability to check for detailed topological relations between regions with a broad boundary,
- The higher levels offer more abstract operators
  That allow users to query uncertain spatial data independent of the underlying geometric data model

2. Metric operators.

- Metric operators provide a more specific description of the object's geometry.
- to measure the relative position of different objects in Terms of distance and direction. Examples include length (arc) and distance (point, point).
- They are used to measure some global properties of Single objects (such as the area, relative size of an object's parts, compactness, And symmetry)

3. **Direction relationship**

- Directional relations are qualitative spatial relations that describe how an object or a region is placed relative to other objects or regions.
- This knowledge is expressed using symbolic (qualitative) and not numerical (quantitative) terms.
- For instance, north, southeast, front, and back-right are directional relations

4. **Ordinal Relationships**

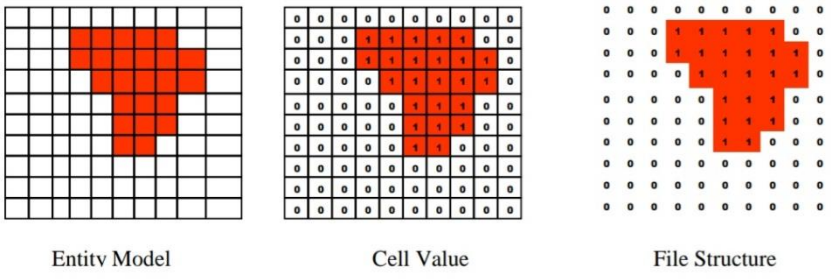
- Ordinal Relationships are represented by relative orders between Objects, which are often described by prepositions such as before and behind.

5. **Projective operators**.

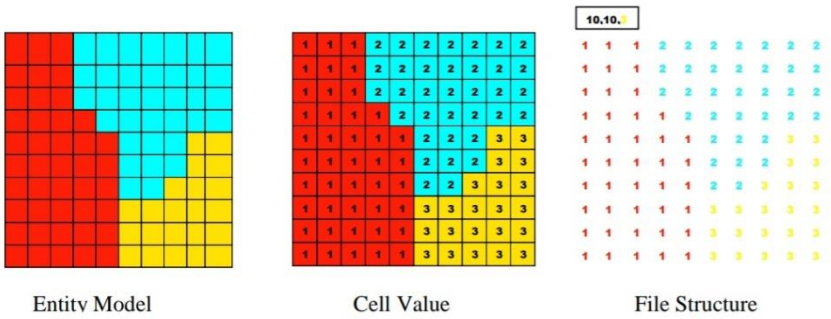
- Projective operators, such as convex hull, are used to Express predicates about the concavity/convexity of objects as well as other Spatial relations

SPATIAL DATA STRUCTURES

Spatial data structures can be classified according to Whether they are used to structure raster or vector data structure.



Entity Model Cell Value File Structure



Entity Model Cell Value File Structure

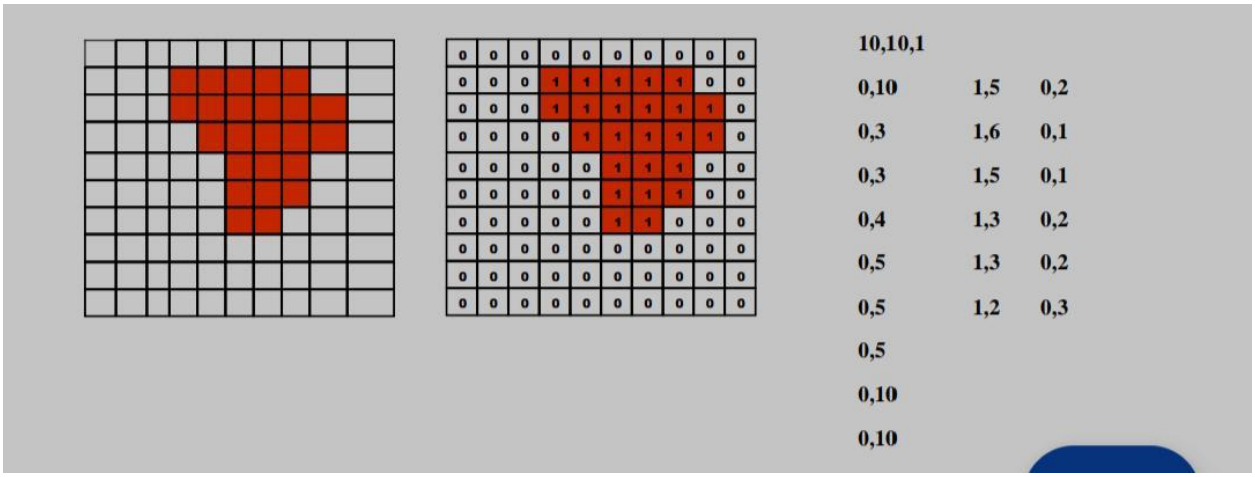
RASTER DATA STRUCTURES

- In the raster data model, each location is represented as a cell.
- The matrix of cells, organized into rows and columns, is called a grid.

- Each row contains a group of cells with values representing a geographic phenomenon.
- The cells in each line of the image are mirrored by an equivalent row of Numbers in the file structure.
- In a simple raster data structure, such as different spatial features must be stored as a separate data
  Layers.
- However, if the entities do not occupy the same geographical location, then it is possible to Store them all in a single layer, with an entity code given to each cell
- One of the major problems with raster data sets their size, because a value must be recorded and stored for each cell in an image.
- Thus, a complex image made up of a mosaic of different Features requires the same amount of storage space as a similar raster map showing the location of A single forest.
- To solve the problem in raster data model the compression or compaction methods Are used for the real-world representation.

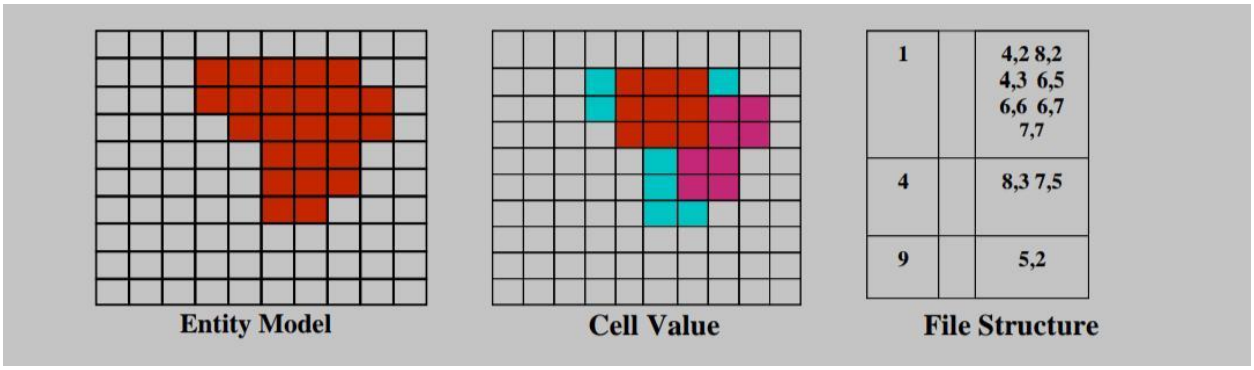
RASTER DATA COMPRESSION METHOD

- Run Length Encoding Method



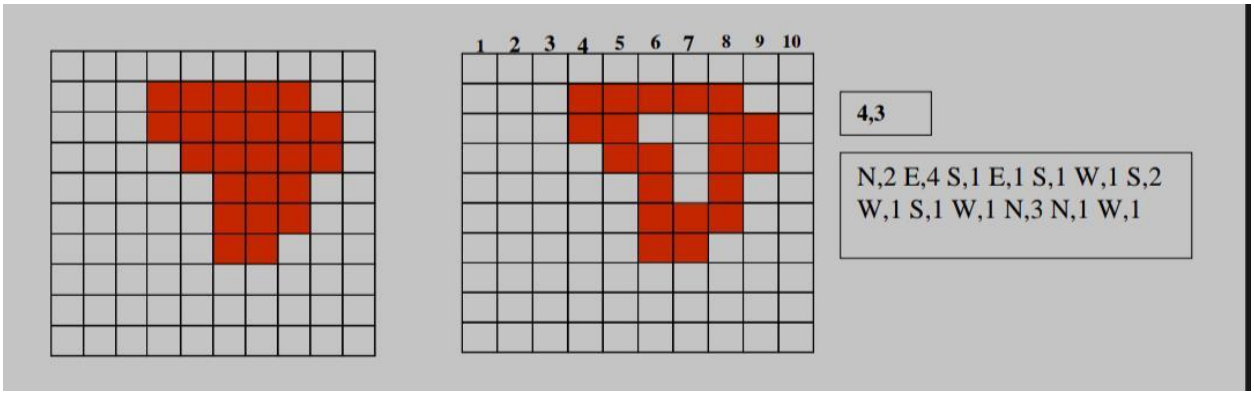
  - It is the raster image compression or compaction method. This technique reduces data volume on a Row-by-row basis.
  - It stores a single value where there are several cells of a given type in a Group, rather than storing for each individual cell.

- Block Coding Method



- This approach extends the run length encoding idea to two dimensions by using a series of square Blocks to store data.

- Chain Coding Method

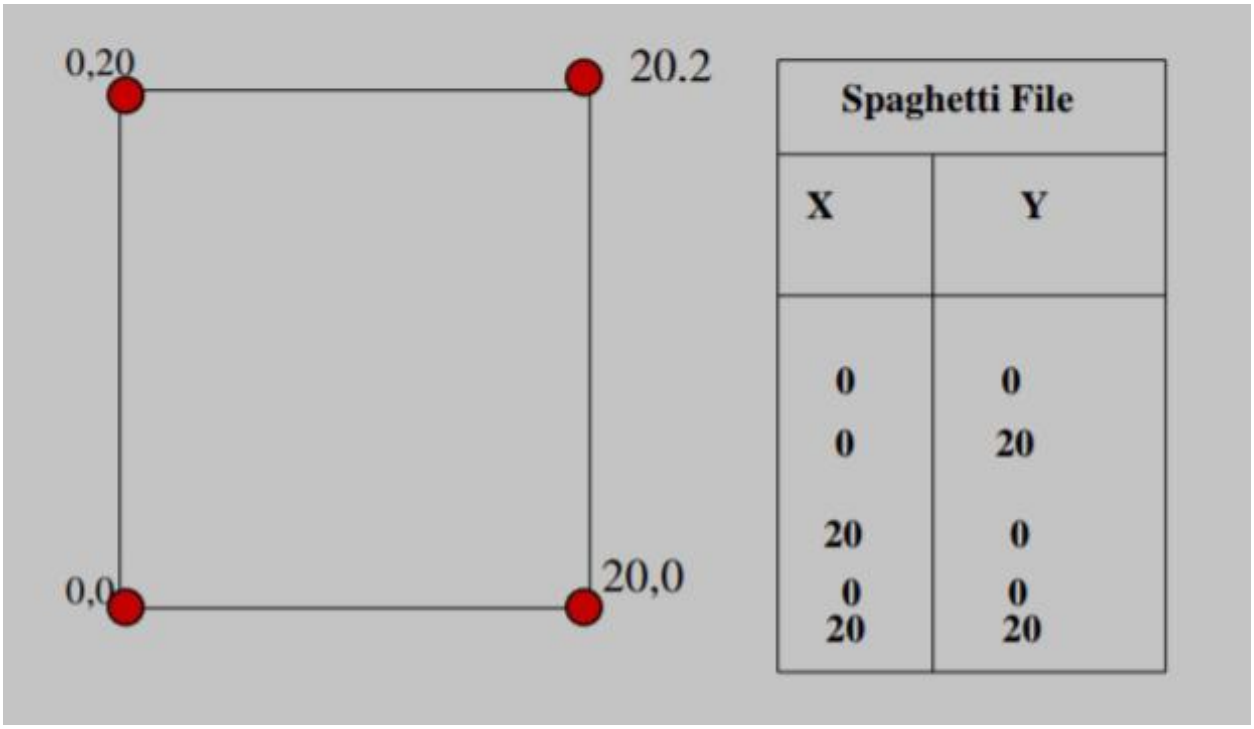


- This method of data can reduce the work by defining the boundary of the entity.
- The boundary is defined as a Sequence of unit cells starting from returning to a given origin.

VECTOR DATA STRUCTURE

- The simplest vector data structure that can be used to produce a graphical image in the computer is a file containing (x, y) coordinate pairs that represent the location of individual Point features.
- In the vector data all points in the data structure must be numbered sequentially
- The vector data structure mainly ensure the following points
  - No node or line segment is duplicated.
  - Line segment and nodes can be referenced to more than one polygon;
  - All polygon has the unique identifiers
  - Polygon can be adequately represented.
- In vector structure topology is concerned with connectivity between entities and not Their physical shape.
- Boundaries are identified through network of arcs, checking polygons for closure, and Linking arcs into polygons.

- The area of polygon can calculate and unique identification numbers are attached.



Spaghetti File	
X	Y
0	0
0	20
20	0
0	0
20	20

SPATIAL ACCESS METHODS

R-TREES

- Instead of dividing space in some manner, it is also Possible to group the objects in some hierarchical
  Organisation based on (a rectangular approximation of) their location.
- The R-tree is a height-balanced tree, which is an extension of the B+-tree for k-dimensions, where k > 1.
- The leaf nodes of this multiway Tree contain entries of the form (I, object-id), where Object-id is a pointer to a data object and I is a Bounding box
- The data object can be of any type: point, polyline, or polygon.
- The internal Nodes contain entries of the form (I, child-pointer), Where child-pointer is a pointer to a child and I is The MBR of that child
- Every node except the root node must be at least half full. Thus, a leaf node That is not the root should contain m entries (I, object-identifier) where M/2<= m
- All leaf nodes are at the same level,
- The root node should have at least Two pointers unless it is a leaf node.
- When inserting a new object, the tree is traversed from the root to a leaf choosing each time the child which needs the least enlargement to enclose the Object.

- If there is still space, then the object is stored in that leaf. Otherwise, the leaf is split into two leaves.
- The entries are distributed among the two Leaves in order to try to minimise the total area of the two leaves.
- A new leaf may cause the parent to Become over-full, so it has to be split also.
- This Process may be repeated up to the root.
- An advantage of the R-tree is that Pointers to complete objects (e.g., polygons) are Stored, so the objects are never fragmented.

THE KD-TREE

- The basic form of the KD-tree stores K-dimensional Points
- Each internal node of the KD-tree contains one point and also corresponds to A rectangular region
- The root of the tree Corresponds to the whole region of interest
- The Rectangular region is divided into two parts by the X-coordinate of the stored point on the odd levels and by the y-coordinate on the even levels in the Tree
- Range Searching in the KD-tree starts at the root, checks whether the stored node (split point) is included in the search range and whether there is overlap with the left or right subtree. For each subtree which Overlaps the search region, the procedure is repeated Until the leaf level is reached.

POINT QUADTREE

- The point quadtree resembles the KD-tree.
- The difference is that the space is Divided into four rectangles instead of two.
- The input points are stored in the internal Nodes of the tree.
- The four different rectangles are Typically referred to as SW (southwest), NW (northwest), SE (southeast), and NE (northeast).
- Searching in the quadtree is very similar to the KD-tree: whenever a point is included in the search Range it is reported and whenever a subtree overlaps with the search range it is traversed.

THE BSP-TREE

- IT is a Data structure that is not based on a rectangular Division of space. Rather, it uses the line segments of the polylines and the edges of the polygons to divide the space in a recursive manner.
- The BSP-tree Reflects this recursive division of space.
- Each time a (sub)space is divided into two subspaces by a so-Called splitting primitive, a corresponding node is Added to the tree

Z ORDERING

- Row ordering simply numbers the cells row by Row, and within each row the points are numbered from left to right
- Ordering techniques are very efficient for **Exact-**match queries for points, but there is quite a Difference in their efficiency for other types of geometric queries, for example range queries.

TEMPORAL DATABASES

OVERVIEW

A Temporal Database is a database with built-in support for handling time sensitive data. (Temporal data)

TEMPORAL DATA

- Temporal Data is the temporary data that is valid only for a prescribed time.
- Temporal data becomes invalid or obsolete after a certain period of time.
- For example, the current temperature of a particular region is temporal data as it keeps on updating and the validity of this temporal data (current temperature) becomes obsolete
- Usually, databases store information only about current state, and not about past states.
- For example, in an employee database if the address or salary of a particular person changes, the database gets updated, the old value is no longer there.
- However, for many applications, it is important to maintain the past or historical values and the time at which the data was updated.
- That is where temporal databases are useful. It stores information about the past, present, and future. Any data that is time dependent is called the temporal data and these are stored in temporal databases.
- Temporal Databases store information about states of the real world across time.
- Temporal Database is a database with built-in support for handling data involving time.
- It stores information relating to past, present, and future time of all events.
- Examples Of Temporal Databases
    - Healthcare Systems: Doctors need the patients' health history for proper diagnosis. Information like the time a vaccination was given or the exact time when fever goes high etc.
    - Insurance Systems: Information about claims, accident history, time when policies are in effect needs to be maintained.
    - Reservation Systems: Date and time of all reservations is important.

TEMPORAL ASPECTS

- **Valid Time:** Time period during which a fact is true in real world, provided to the system.
- **Transaction Time:** Time period during which a fact is stored in the database, based on transaction serialization order
- **Decision Time:** Decision time in the temporal database is the time at which the decision is made about the fact.

TYPES OF TEMPORAL RELATION

- **Uni-Temporal Relation:** The relation which is associated with valid or transaction time is called Uni-Temporal relation. It is related to only one time.
- **Bi-Temporal Relation:** The relation which is associated with both valid time and transaction time is called a Bi-Temporal relation. Valid time has two parts namely start time and end time, similar in the case of transaction time.
- **Tri-Temporal Relation:** The relation which is associated with three aspects of time namely Valid time, Transaction time, and Decision time called as Tri-Temporal relation.

CHALLENGES OF TEMPORAL DATABASES

- Data Storage: In temporal databases, each version of the data needs to be stored separately. As a result, storing the data in temporal databases requires more storage as compared to storing data in non-temporal databases.
- Schema Design: The temporal database schema must accommodate the time dimension. Creating such a schema is more difficult than creating a schema for non-temporal databases.
- Query Processing: Processing the query in temporal databases is slower than processing the query in non-temporal databases due to the additional complexity of managing temporal data.

ACTIVE DATABASES

The model that has been used to specify active database rules is referred to as the Event-Condition-Action (ECA) model. A rule in the ECA model has three components:

1. **The event(s) that triggers the rule:** These events are usually database update Operations that are explicitly applied to the database. However, in the general Model, they could also be temporal events2 or other kinds of external events.
2. **The condition that determines whether the rule action should be executed:** Once the triggering event has occurred, an optional condition may be evaluated. If no condition is specified, the action will be executed once the event Occurs. If a condition is specified, it is first evaluated, and only if it evaluates to true will the rule action be executed.

3. **The action to be taken:** The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will Be automatically executed.

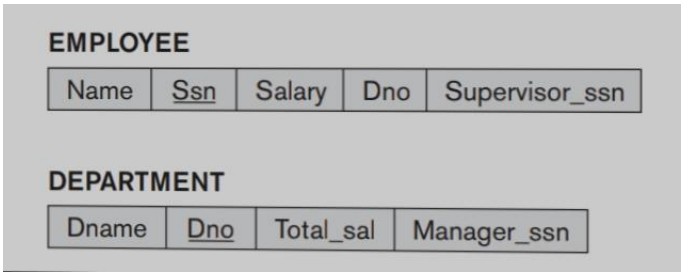
FEATURES OF ACTIVE DATABASE:

- It possesses all the concepts of a conventional database i.e., data modelling facilities, query language etc.
- It supports all the functions of a traditional database like data definition, data manipulation, storage management etc.
- It detects event occurrence.
- It must be able to evaluate conditions and to execute actions.

ADVANTAGES:

- Enhances traditional database functionalities with powerful rule processing capabilities.
- Avoids redundancy of checking and repair operations.
- Suitable platform for building large and efficient knowledge base and expert systems.

EXAMPLE



Notice that

- We Assume that NULL is allowed for Dno, indicating that an employee may be temporarily unassigned to any department.
- Each department has a name (Dname), nnumberl (Dno), the total salary of all employees assigned to the department (Total_sal), and a Manager (Manager_ssn, which is a foreign key to EMPLOYEE).
- the Total_sal attribute is really a derived attribute, whose value should be the sum of the salaries of all employees who are assigned to the particular department.

Maintaining the correct value of such a derived attribute can be done via an Active rule.

Events

First, we have to determine the events that may cause a change in the Value of Total_sal, which are as follows:

1. Inserting (one or more) new employee tuples
2. Changing the salary of (one or more) existing employees
3. Changing the assignment of existing employees from one department to Another
4. Deleting (one or more) employee tuples

Condition

In the case of event 1,

- we only need to recompute Total_sal if the new employee is Immediately assigned to a department—that is, if the value of the Dno attribute for the new employee tuple is not NULL (assuming NULL is allowed for Dno). Hence, this Would be the condition to be checked

A similar condition could be checked for Event 2 and 4

- to determine whether the employee whose salary is changed (or who Is being deleted) is currently assigned to a department.

For event 3,

- we will always Execute an action to maintain the value of Total_sal correctly, so no condition is Needed (the action is always executed).

Action

- The action for events 1, 2, and 4 is to automatically update the value of Total_sal for the employee's department to reflect the newly inserted, updated, or deleted Employee's salary.
- In the case of event 3, a twofold action is needed: one to update The Total_sal of the employee's old department and the other to update the Total_sal of the employee's new department.

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.
- 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.

- In a deductive database system, we typically specify rules through a declarative language—a language in which we specify what to achieve rather than how to achieve It.
- An inference engine (or deduction mechanism) within the system can deduce new facts from the database by interpreting these rules.
- The deductive database work Based on logic has used Prolong as a starting point. A variation of Prolog called Data log
- A deductive database uses two main types of specifications: facts and rules.
    - **Facts** are Specified in a manner similar to the way relations are specified, except that it is not Necessary to include the attribute names. In A deductive database, the meaning of an attribute value in a tuple is determined Solely by its position within the tuple.
    - **Rules** specify virtual relations that are not actually stored but that can be Formed from the facts by applying inference mechanisms based on the rule specifications.

PROLOG/DATA LOG NOTATION

- A predicate has an implicit meaning, which is suggested by the predicate Name, and a fixed number of arguments.

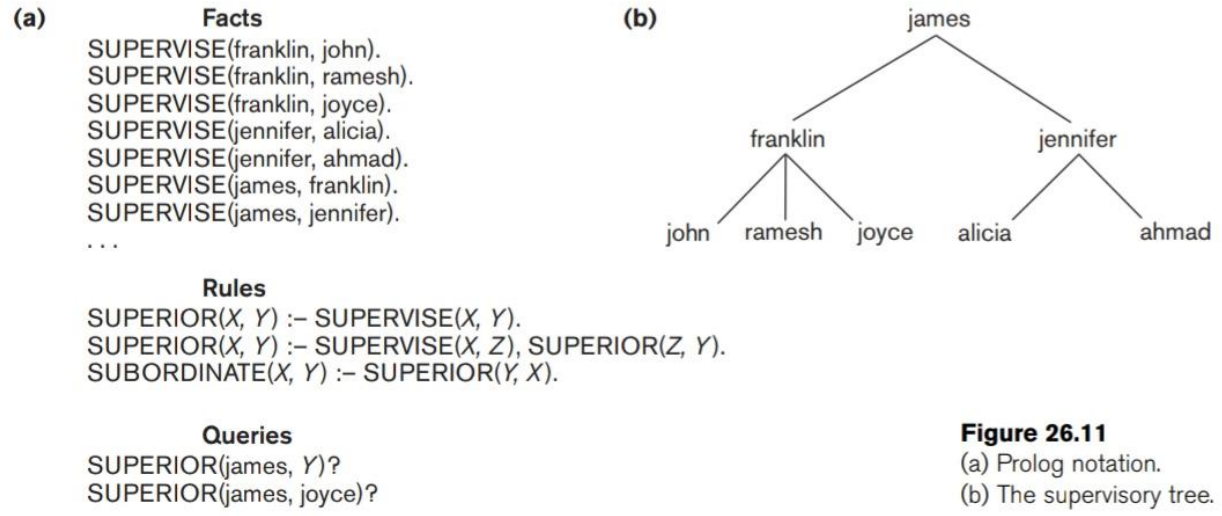


Figure 26.11
(a) Prolog notation.
(b) The supervisory tree.

- There are three predicate names: Supervise, superior, and subordinate.
- The SUPERVISE predicate is defined via a set of
  Facts, each of which has two arguments: a supervisor name, followed by the name of A direct supervisee (subordinate) of that supervisor.
  SUPERVISE (Supervisor, Supervisee)
- Thus, SUPERVISE (X, Y) states the fact that X supervises Y.
- Consider the definition of the predicate SUPERIOR in, whose first Argument is an employee name and whose second argument is an employee who is Either a direct or an indirect subordinate of the first employee.

- By indirect subordinate, we mean the subordinate of some subordinate down to any number of levels.
- Thus SUPERIOR (X, Y) stands for the fact that X is a superior of Y through direct or Indirect supervision.
- We can write two rules that together specify the meaning of the new predicate.
    - The first rule under Rules in the figure states that for every value of X And Y, if SUPERVISE (X, Y)—the rule body—is true, then SUPERIOR (X, Y)—the Rule head—is also true, since Y would be a direct subordinate of X (at one level Down). This rule can be used to generate all direct superior/subordinate relation-Ships from the facts that define the SUPERVISE predicate.
    - The second recursive rule States that if SUPERVISE (X, Z) and SUPERIOR (Z, Y) are both true, then SUPERIOR (X, Y) is also true. This is an example of a recursive rule

DATA LOG NOTATION

- In Data log, as in other logic-based languages, a program is built from basic objects Called atomic formulas.
- It is customary to define the syntax of logic-based languages by describing the syntax of atomic formulas and identifying how they can be Combined to form a program

RECURSIVE QUERIES IN SQL

- Recursive queries in SQL refer to queries that can be defined recursively, allowing for the retrieval of hierarchical or recursive data structures.
- These queries are particularly useful when working with data that has a self-referential or hierarchical relationship.
- Recursive queries are supported by specific features in SQL, such as recursive common table expressions (CTEs).

Here is a detailed explanation of recursive queries in SQL:

FORMS OF RECURSIVE QUERIES

1. **Recursive Data Structures:**
    - Recursive queries are typically used when dealing with data structures that exhibit a hierarchical or self-referential relationship.
    - Examples of such data structures include organizational hierarchies, file systems, network topologies, and nested comments or replies.
2. **Recursive Common Table Expressions (CTEs):**
    - Recursive queries in SQL are implemented using recursive common table expressions (CTEs).

- A CTE is a named temporary result set that can be self-referencing, allowing for recursion. It consists of two parts: the base case (initialization) and the recursive case.

3. **Base Case:**
   - The base case is the non-recursive part of the CTE, which serves as the starting point for the recursion.
   - It defines the initial set of rows or records to be included in the result. The base case is typically a regular SELECT statement without any recursive references.
4. **Recursive Case:**
   - The recursive case is the part of the CTE that defines how to derive additional rows from the previously generated rows.
   - It contains a SELECT statement that refers to the CTE itself, creating a recursive loop. The recursive SELECT statement joins the CTE with other tables or itself, applying specific conditions to establish the recursive relationship.
5. **Anchor Member and Recursive Member:**
   - In a recursive CTE, each iteration of the recursion has two components: the anchor member and the recursive member.
   - The anchor member represents the base case, while the recursive member represents the recursive case.
   - The anchor member provides the initial set of rows, and the recursive member generates additional rows based on the previous iteration's results.
6. **UNION ALL Operator:**
   - The recursive member of the CTE is typically combined with the anchor member using the UNION ALL operator.
   - UNION ALL allows the results from the anchor member and the recursive member to be combined into a single result set. It ensures that the recursive query includes all the generated rows in each iteration.
7. **Recursive Termination Condition:**
   - Recursive queries require a termination condition to prevent infinite recursion.
   - The termination condition is defined in the recursive SELECT statement and serves as a stopping criterion for the recursion.
   - It specifies when the recursion should stop and no further iterations should occur.
   - Typically, it involves a condition that evaluates to false or a limit on the number of iterations.
8. **Result Retrieval:**
   - Once the recursive CTE is defined, the result of the query is obtained by selecting from the CTE itself.
   - The result set contains all the rows generated during the recursive iterations.

BENEFITS OF RECURSIVE QUERIES:

1. **Hierarchical Data Retrieval:**
   - Recursive queries enable the retrieval of hierarchical or recursive data structures.
   - They provide a convenient way to traverse and query data that has a self-referential relationship, such as parent-child or ancestor-descendant relationships.

2. **Flexibility in Querying:**
   - Recursive queries allow for flexible and powerful querying of hierarchical data.
   - They can be used to retrieve specific levels of the hierarchy, find paths between nodes, calculate aggregations at different levels, and perform other complex operations on hierarchical structures.
3. **Code Simplicity and Readability:**
   - Recursive queries provide a concise and readable way to express recursive relationships in SQL.
   - They eliminate the need for complex and repetitive code structures, resulting in more maintainable and understandable queries.
4. **Efficiency:**
   - Recursive queries in SQL are optimized for performance. The underlying database engine employs various techniques, such as memorization and query optimization, to ensure efficient execution of recursive queries.
5. **Performance Optimization:**
   - Recursive queries can often be more efficient than alternative methods of traversing hierarchical data, as they leverage the power of the database engine's optimization capabilities. With proper indexing and query optimization, recursive queries can provide fast and scalable results.

EXAMPLES

- Recursive joins are sometimes also called "fixed-point joins."
- A recursive query is a powerful feature that allows us to query hierarchical data which are used in relational databases.
- This involves joining a set with itself an arbitrary number of times.
- A recursive query is usually defined by the anchor part and the recursive part.
- In SQL Recursive joins are implemented with recursive common table expressions.
- Recursive common table expression (CTEs) is a way to reference a query over and over again.

Step 1:

- First, we create a database of employees, Where Common Table Expression of the company for its Employee Id, Employee name, Employee age.

Query:

```
CREATE TABLE employees (
Id serial,
Name varchar (20),
Age int
);
```

Step 2:

- In this step insert values into an employee table.

Query:

```
INSERT INTO employees VALUES (1, 'Ankit', 32);
INSERT INTO employees VALUES (2, 'Ayush', 31);
INSERT INTO employees VALUES (3, 'Piyush', 42);
INSERT INTO employees VALUES (4, 'Ramesh', 31);
INSERT INTO employees VALUES (5, 'Rohan', 29);
INSERT INTO employees VALUES (6, 'Harry', 28);
INSERT INTO employees VALUES (7, 'Rohit', 32);
INSERT INTO employees VALUES (8, 'Gogi', 32);
INSERT INTO employees VALUES (9, 'Tapu', 33);
INSERT INTO employees VALUES (10, 'Sonu', 40);
```

Step 3:

- A statement that gives all the reports that roll up into a certain organization within the company.
- A CTE is defined using a WITH statement, followed by a table expression definition.
- The AS command is used to rename a column or table with an alias.
- A recursive CTE must contain a UNION statement and be recursive.

Query:

```
WITH RECURSIVE manager tree AS (

    SELECT id, name, age

    FROM employees

    WHERE id = 1

    UNION ALL

    SELECT e.id, e.name, e.age

    FROM employees e

    INNER JOIN managertree mtree ON mtree.id = e.age

   )
```

Step 4:

- To check the recursive join data we use the following query.

Query:

```
SELECT * FROM managertree;
```

Output:

id	name	age
1	Ankit	32
2	Ayush	31
3	Piyush	42
4	Ramesh	31
5	Rohan	29
6	Harry	28
7	Rohit	32
8	Gogi	32
9	Tapu	33
10	Sonu	40

MOBILE DATABASES

- A Mobile database is a database that can be connected to a mobile computing device over a mobile network (or wireless network).
- Here the client and the server have wireless connections.
- It will be applicable on different-different devices like android based mobile databases, iOS based mobile databases, etc.

FEATURES OF MOBILE DATABASE:

- Mobile databases resided on mobile devices.
- As the use of laptops, mobile and PDAs is increasing to reside in the mobile system.
- Mobile databases are physically separate from the central database server.
- Mobile databases are capable of communicating with a central database server or other mobile clients from remote sites.
- With the help of a mobile database, mobile users must be able to work without a wireless connection due to poor or even non-existent connections (disconnected).
- A cache is maintained to hold frequent and transactions so that they are not lost due to connection failure.
- A mobile database is used to analyse and manipulate data on mobile devices.

Mobile Database typically involves three parties:

Fixed Hosts

- It performs the transactions and data management functions with the help of database servers.

Mobiles Units

- These are portable computers that move around a geographical region that includes the cellular network that these units use to communicate to base stations.

Base Stations

- These are two-way radios installation in fixed locations, that pass communication with the mobile units to and from the fixed hosts.

LOCATION AND HANDOFF MANAGEMENT

- In cellular systems a mobile unit is free to move around within the entire area of coverage.
- Its movement is random and therefore its geographical location is unpredictable.
- This situation makes it necessary to locate the mobile unit.
- Thus, the entire process of the mobility management component of the cellular system is responsible for two tasks:
    - **location management**- identification of the current geographical location or current point of attachment of a mobile unit which is required by the MSC (Mobile Switching Centre) to route the call.
    - **handoff**- transferring (handing off) the current (active) communication session to the next base station, which seamlessly resumes the session using its own set of channels.
- The current point of location of a subscriber (mobile unit) is expressed in Terms of the cell or the base station to which it is presently connected. The mobile units (called and calling subscribers) can continue to talk and move around in the irrespective cells; but as soon as both or any one of the units moves to a different cell, the location management procedure is invoked to identify the new location.
- The location management performs three fundamental tasks:
    - **Location update**-is initiated by the mobile unit; the current location of the unit is recorded in HLR and VLR databases.
    - **Location lookup**- a database search to obtain the current location of the mobile unit.
    - **Paging** -the system informs the caller the location of the called unit in terms of its current base station.
- A mobile unit can freely move around
    - In **active mode**, the mobile actively communicates with other subscriber, and it may continue to move within the cell or may encounter a handoff which May interrupt the communication. It is the task of the location manager to find the new location and resume the communication.

- In doze **mode** a mobile unit does not actively communicate with other subscribers but continues to listen to the base station and monitors the signal levels around it
- In **Power down** mode the unit is not functional at all.

- When it moves to a different cell in doze or power down modes, then it is neither possible nor necessary for the location manager to find the location.

MOBILE TRANSACTION MODELS

- The conventional ACID transaction model was unable to satisfactorily manage mobile data processing tasks.
- Some of the important reasons were: the presence of handoff, which is unpredictable; the presence of doze mode, disconnected mode, and forced disconnection; lack of necessary resources such as memory, and wireless channels; presence of location dependent data; etc.
- To manage data processing in the presence of these new issues, a more powerful transaction model or ACID transaction execution model that can handle mobility during data processing was highly desirable.

TYPES OF DATABASES

1. **HiCoMo**:
    - High Commit Mobile Transaction Model Although it has been presented as a mobile transaction model, in reality it is a mobile transaction execution model.
    - The execution model is mainly for processing aggregate data stored in a data warehouse which resides in mobile units.
    - Since the data warehouse resides in mobile units, HiCoMo transactions are always initiated on mobile units where they are processed in a disconnected mode.
    - As a result, transaction commitments are quite fast. The results of these transactions are then installed in the database upon reconnection.
    - The base database resides on the fixed network. It is manipulated by transactions called base or source transactions. T
    - here transactions are initiated at the fixed network.
    - Transaction which are initiated and processed at mobile units are called HiCoMo.
    - Since HiCoMo transactions do specialized processing, it is based on the following assumptions: The data warehouse stores aggregate data of the following types: average, sum, minimum, and maximum. Operations such as subtraction, addition, and multiplication are allowed with some constraints.
2. **Kangaroo Transactions**,
    - Kangaroo Transactions which create a sub transaction on every MSS that the user passes by, establishing a link between them so as to move the data as the user moves.
    - It can use compensations, if necessary, in case of failures.
3. **Moflex Transaction Model**

- A mobile transaction model called Moflex, which is based on a flexible transaction model. The structure of a Moflex has 7 components and can be defined as Moflex transaction

  T = {M, S, F, D, H, J, G)
- It allows for the definition of parameters that specify, in a flexible way, the behaviour of the transactions.
- A transaction is made up of sub transactions. It is possible to define the behaviour these transactions will follow, such as the success or failure dependencies they must maintain with other sub transactions, conditions regarding the geographic location, etc.
- When a transaction faces a handoff, it can behave as specified by the user through a series of rules.

CONCURRENCY

Concurrency in mobile databases refers to the ability to handle multiple concurrent transactions or operations in a mobile computing environment. Mobile databases face unique challenges related to concurrency due to the mobility of devices and intermittent connectivity. Here are some key aspects of concurrency in mobile databases:

1. Isolation and Consistency: Concurrency control mechanisms, such as locking or optimistic concurrency control, are employed to ensure isolation and consistency of data in mobile databases. These mechanisms prevent conflicts and ensure that concurrent transactions do not interfere with each other or produce inconsistent results.

2. Resource Contentions: Mobile devices typically have limited resources, including processing power, memory, and network bandwidth. Concurrent transactions may compete for these resources, leading to contention and potential performance degradation. Efficient resource management techniques are necessary to mitigate contention and optimize resource utilization.

3. Conflict Resolution: In mobile environments, conflicts can arise when multiple devices or users attempt to access or modify the same data simultaneously. Conflict resolution strategies, such as timestamp-based ordering or conflict detection and resolution algorithms, are employed to resolve conflicts and maintain data integrity.

4. Replication and Synchronization: Mobile databases often employ replication strategies to improve data availability and performance. Replicas of the database can be maintained on both the mobile device and a central server. Concurrency control mechanisms must ensure proper synchronization between replicas to maintain consistency and avoid conflicts during replication.

5. Asynchronous Operations: Mobile devices may experience intermittent network connectivity, leading to delays or interruptions in communication with the central database server. Asynchronous operations allow mobile devices to continue executing transactions or operations

locally without being blocked by network unavailability. Proper concurrency control mechanisms should handle these scenarios effectively.

6. Handoff Management: Mobile devices frequently change their network connectivity or switch between different access points. During handoffs, ongoing transactions must be managed to ensure continuity and consistency. Concurrency control mechanisms should account for handoff scenarios and provide mechanisms to handle them seamlessly.

7. Deadlock Detection and Prevention: Mobile databases may encounter deadlock situations when multiple transactions are waiting for resources held by others, resulting in a cyclic dependency. Deadlock detection and prevention mechanisms, such as deadlock detection algorithms or timeout-based techniques, are employed to identify and resolve deadlocks to maintain system responsiveness.

8. Conflict-Aware Scheduling: Concurrency control mechanisms in mobile databases need to consider the specific characteristics of mobile environments. Conflict-aware scheduling algorithms take into account factors such as network availability, battery levels, and user preferences to prioritize transactions and optimize resource usage.

Efficient concurrency management in mobile databases is crucial to ensure data consistency, performance, and responsiveness in the face of limited resources, intermittent connectivity, and dynamic mobility. It requires a combination of concurrency control techniques, replication strategies, conflict resolution mechanisms, and efficient resource management approaches tailored to the unique challenges of mobile computing.

TRANSACTIONS COMMIT PROTOCOL

Transaction commit protocols in mobile databases ensure the atomicity, consistency, isolation, and durability (ACID) properties of transactions when committing changes to the database. Due to the unique characteristics of mobile environments, such as limited connectivity, intermittent network availability, and mobility of devices, specific protocols are employed to handle transaction commits effectively. Here are some commonly used transactions commit protocols in mobile databases:

1. Two-Phase Commit (2PC): The Two-Phase Commit protocol is a widely used protocol for transaction commit coordination. It involves a coordinator, typically a central server, and multiple participants, which can be mobile devices or servers. The protocol consists of two phases: prepare and commit. In the prepare phase, the coordinator asks all participants if they are ready to commit. If all participants respond positively, the coordinator sends a commit request in the commit phase. If any participant cannot commit, the coordinator sends an abort request to all participants. This protocol ensures that either all participants commit or none commit, maintaining transaction atomicity.

2. Three-Phase Commit (3PC): The Three-Phase Commit protocol is an extension of the Two-Phase Commit protocol that adds an extra phase called the pre-commit phase. The pre-commit phase allows participants to agree on the decision to commit before the actual commit request is sent. This additional phase helps to handle failures more efficiently, as it enables participants to indicate their preparedness to commit or abort in advance. 3PC reduces the blocking time experienced by participants in 2PC, which is particularly beneficial in mobile environments with potential connectivity issues.

3. Optimistic Commit Protocol: The Optimistic Commit protocol assumes that conflicts between concurrent transactions are rare. In this protocol, transactions are allowed to execute without acquiring locks or blocking other transactions. At the commit phase, conflicts are checked and resolved. If conflicts occur, one or more transactions may need to be rolled back and retried. Optimistic Commit protocols are useful in scenarios where conflicts are infrequent, as they can provide better concurrency and reduced resource contention.

4. Deferred Update Commit Protocol: The Deferred Update Commit protocol is designed for mobile environments with limited connectivity or intermittent network availability. In this protocol, updates made by a transaction are not immediately sent to the central server for commit. Instead, they are stored locally on the mobile device until network connectivity is available. Once connectivity is restored, the updates are sent to the server for commit. This protocol reduces the reliance on continuous network connectivity and allows transactions to be completed locally, improving responsiveness and availability.

5. Batched Commit Protocol: The Batched Commit protocol aims to reduce the number of communication rounds between the mobile device and the central server during transaction commits. Instead of committing each transaction individually, multiple transactions are batched together and sent to the server in a single communication round. This reduces the overhead of network communication and improves efficiency, especially in situations with limited bandwidth or high latency.

Each of these transaction commit protocols has its advantages and trade-offs in terms of performance, fault tolerance, and resource utilization. The choice of a specific protocol depends on the characteristics of the mobile database environment, the level of network connectivity, the expected frequency of conflicts, and the desired level of transactional consistency and availability.

MULTIMEDIA DATABASES

- Multimedia databases provide features that allow users to store and query different Types of multimedia information, which includes images (such as photos or drawings), video clips (such as movies, newsreels, or home videos), audio clips (such as Songs, phone messages, or speeches), and documents (such as books or articles).
- Content-based retrieval
    - One may want to locate all Video clips in a video database that include a certain person, say Michael Jackson.

    - One may also want to retrieve video clips based on certain activities included in Them, such as video clips where a soccer goal is scored by a certain player or team.
    - The above types of queries are referred to as content-based retrieval, because the Multimedia source is being retrieved based on its containing certain objects or Activities
- Hence, a multimedia database must use some model to organize and Index the multimedia sources based on their contents.
- Identifying the contents of Multimedia sources is a difficult and time-consuming task. There are two main Approaches.
    - Automatic analysis – This approach uses Different techniques depending on the type of multimedia source
    - Manual identification – This approach can be applied to all multimedia Sources, but it requires a manual preprocessing phase where a person has to scan Each multimedia source to identify and catalogue the objects and activities it contains so that they can be used to index the sources.

- **Image**
    - An image is typically stored either in raw form as a set of pixel or cell values, or in Compressed form to save space.
    - Hence, each image can be represented by an m by n grid of cells.
    - Each cell contains a pixel value that describes the cell content.
    - In black-and-white images, Pixels can be one bit.
    - In grey scale or colour images, a pixel is multiple bits. Because Images may require large amounts of space, they are often stored in compressed Form.
    - To identify objects of interest in an image, the image is typically divided into homogeneous segments using a homogeneity predicate.
    - For example, in a colour image, adjacent cells that have similar pixel values are grouped into a segment.
    - The homogeneity Predicate defines conditions for automatically grouping those cells.
    - Segmentation And compression can hence identify the main characteristics of an image.
    - A typical image database query would be to find images in the database that are Similar to a given image. The given image could be an isolated segment that contains, say, a pattern of interest, and the query is to locate other images that contain That same pattern.
    - There are two main techniques for this type of search.
    - The first Approach uses a **distance function** to compare the given image with the stored
      Images and their segments.
    - The second approach, called the Transformation approach, measures image similarity by having a small number of
      Transformations that can change one image's cells to match the other image. Transformations include rotations, translations, and scaling.

- **Video**
    - A video source is typically represented as a sequence of frames, where each frame is a still image.
    - However, rather than identifying the objects and activities in every Individual frame, the video is divided into video segments, where each segment Comprises a sequence of contiguous frames that includes the same objects/activities.
    - Each segment is identified by its starting and ending frames.
    - The objects and activities identified in each video segment can be used to index the segments.
    - The Index includes both objects, such as persons, houses, and cars, as well as activities,
      Such as a person delivering a speech or two people talking.
    - Videos are also often Compressed using standards such as MPEG**.**
- **Audio sources**
    - Audio sources include stored recorded messages, such as speeches, class presentations, or even surveillance recordings of phone messages or conversations by law enforcement.
    - Here, discrete transforms can be used to identify the main characteristics of a certain person's voice in order to have similarity-based indexing and Retrieval.
- **Text document**
    - A text/document source is basically the full text of some article, book, or magazine.
    - These sources are typically indexed by identifying the keywords that appear in the text and their relative frequencies.
    - However, filler words or common words called Stop words are eliminated from the process.