

Overview on concept of DBMS, RDBMS and SDBMS for geo-data handling

Dharmendra Kumar

Topics we will learn 1/3

- What is Data?
- Database and Database Management System
- Types of Databases:
 - Centralized
 - Distributed
 - Relational
 - NoSQL
 - Cloud AND Hierarchical
 - OTHERS
- Characteristics of DBMS
- Advantages of DBMS
- Disadvantages of DBMS

What is Data?

- Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.
- Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.
- In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

Database and Database Management System

- A **database** is an organized collection of data, so that it can be easily accessed and managed.
- You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
- **Database handlers** create a database in such a way that only one set of software program provides access of data to all the users.
- The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.
- There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.
- There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.
- Modern databases are managed by the database management system (DBMS).
- **SQL** or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

Types of Databases-Centralized

- It is the type of database that stores data at a centralized database system. It comforts the users to access the stored data from different locations through several applications. These applications contain the authentication process to let users access data securely. An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.
- Advantages of Centralized Database
 - It has decreased the risk of data management, i.e., manipulation of data will not affect the core data.
 - Data consistency is maintained as it manages data in a central repository.
 - It provides better data quality, which enables organizations to establish data standards.
 - It is less costly because fewer vendors are required to handle the data sets.
- Disadvantages of Centralized Database
 - The size of the centralized database is large, which increases the response time for fetching the data.
 - It is not easy to update such an extensive database system.
 - If any server failure occurs, entire data will be lost, which could be a huge loss.

Types of Databases-Distributed

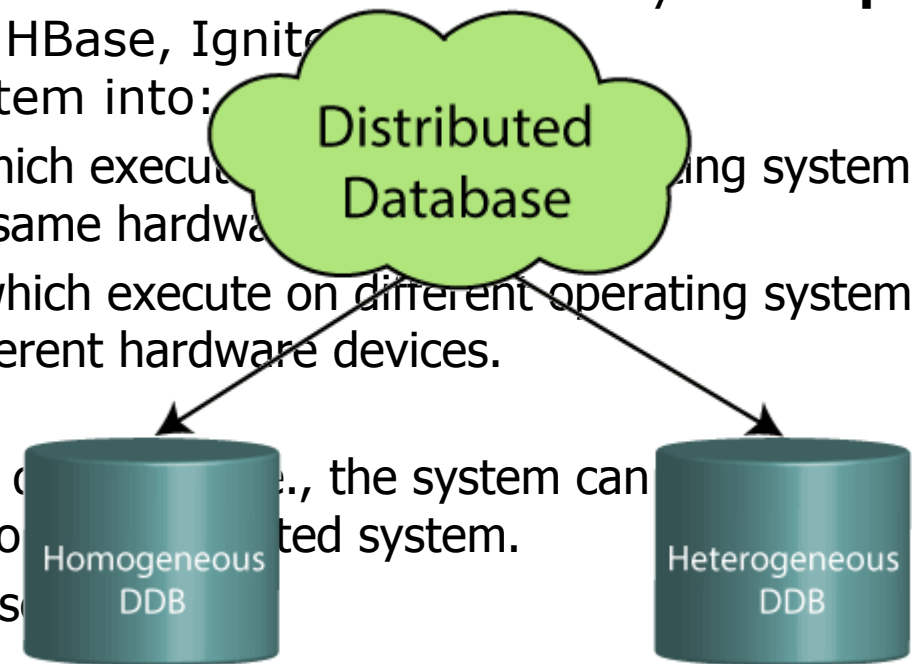
Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization. These database systems are connected via communication links. Such links help the end-users to access the data easily. **Examples** of the Distributed database are Apache Cassandra, HBase, Ignite.

We can further divide a distributed database system into:

- Homogeneous DDB: Those database systems which execute on the same operating system and use the same application process and carry the same hardware.
- Heterogeneous DDB: Those database systems which execute on different operating systems under different application procedures, and carries different hardware devices.

Advantages of Distributed Database

- Modular development is possible in a distributed database system, the system can be expanded by including new computers and connecting them to the existing system.
- One server failure will not affect the entire data system.



Types of Databases-Relational

- This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). A relational database uses SQL for storing, manipulating, as well as maintaining the data. E.F. Codd invented the database in 1970. Each table in the database carries a key that makes the data unique from others. **Examples** of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.
- Properties of Relational Database: There are following four commonly known properties of a relational model known as ACID properties, where:
 - **A means Atomicity:** This ensures the data operation will complete either with success or with failure. It follows the 'all or nothing' strategy. For example, a transaction will either be committed or will abort.
 - **C means Consistency:** If we perform any operation over the data, its value before and after the operation should be preserved. For example, the account balance before and after the transaction should be correct, i.e., it should remain conserved.
 - **I means Isolation:** There can be concurrent users for accessing data at the same time from the database. Thus, isolation between the data should remain isolated. For example, when multiple transactions occur at the same time, one transaction effects should not be visible to the other transactions in the database.
 - **D means Durability:** It ensures that once it completes the operation and commits the data, data changes should remain permanent.

Types of Databases-NoSQL

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways. It came into existence when the demand for building modern applications increased. Thus, NoSQL presented a wide variety of database technologies in response to the demands. We can further divide a NoSQL database into the following four types:

Key-value storage: It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.

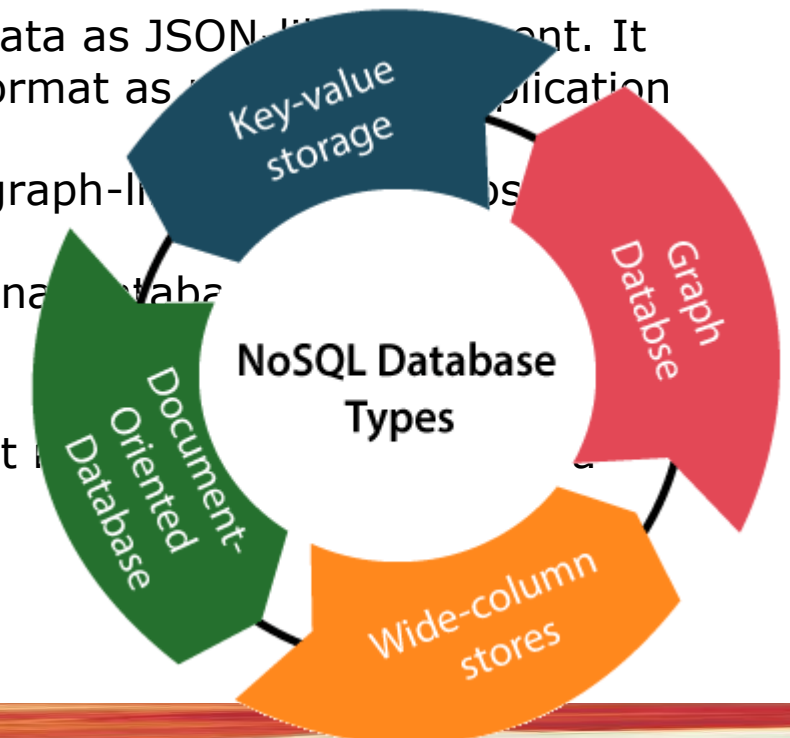
2.Document-oriented Database: A type of database used to store data as JSON. It helps developers in storing data by using the same document-model format as application code.

3.Graph Databases: It is used for storing vast amounts of data in a graph-like structure. commonly, social networking websites use the graph database.

4.Wide-column stores: It is similar to the data represented in relational database stored in large columns together, instead of storing in rows.

Advantages of NoSQL Database

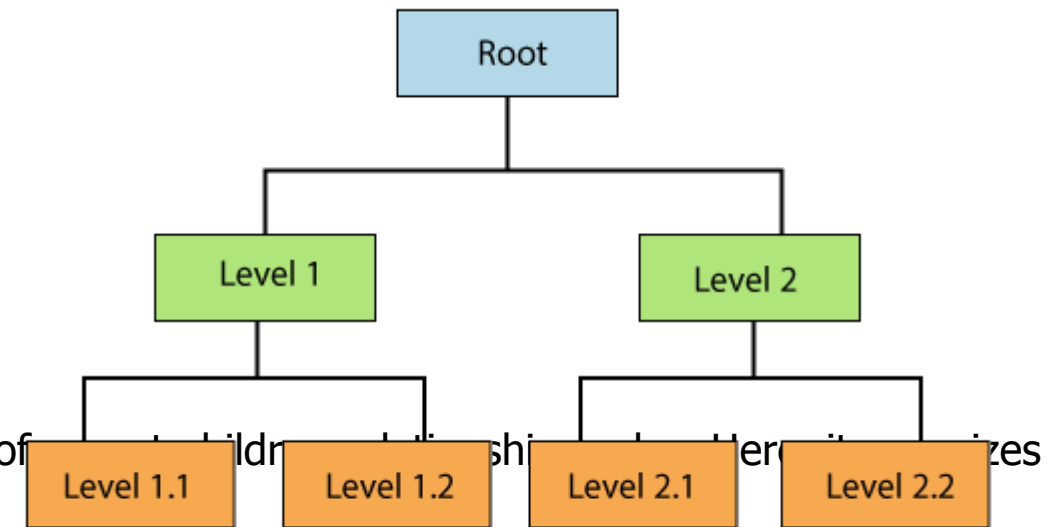
- It enables good productivity in the application development as it is not in structured format.
- It is a better option for managing and handling large data sets.
- It provides high scalability.
- Users can quickly access data from the database through key-value.



Types of Databases- Cloud and Hierarchical

■ Cloud:

- A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database. There are numerous cloud platforms, but the best options are:
 - Amazon Web Services(AWS)
 - Microsoft Azure
 - Kamatera
 - PhonixNAP
 - ScienceSoft
 - Google Cloud SQL, etc.



■ Hierarchical:

- It is the type of database that stores data in the form of hierarchical relationship. It organizes data in a tree-like structure.
- Data get stored in the form of records that are connected via links. Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.

Hierarchical Database

Types of Databases- others

■ Object-oriented Databases

- The type of database that uses the object-based data model approach for storing data in the database system. The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

■ Network Databases

- It is the database that typically follows the network data model. Here, the representation of data is in the form of nodes connected via links between them. Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Relational Data Model

Topics we will learn 2/3

- A Brief History of Relational Data Models
- What is Relational Model
- Relational Model Concepts
- Relational Integrity constraints
- Operations in Relational Model
- Best Practices for creating a Relational Model
- Advantages of using Relational model
- Disadvantages of using Relational model
- Sample Queries

A Brief History of Relational Data Models

- Introduced by Dr. E. F. Codd's seminal paper in 1970.
- Dr. Codd give set of rules for RDBMS
- The ER model is used for conceptual modeling purpose
- Any RDBMS consists of:
 - A collection of tables
 - Rules (constraints) for how tables are arranged
 - A set of operators to act on the tables

- Data Types
- Relationship between table
- SQL Queries- Example (DDL, DML, DCL)
- Indexing Basics

What is Relational Model

- The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.
- Some popular Relational Database management systems are:
 - DB2 and Informix Dynamic Server - IBM
 - Oracle and RDB – Oracle
 - SQL Server and Access - Microsoft

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME, etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

Table also called Relation

Primary Key

Domain
Ex: NOT NULL

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Tuple OR Row
Total # of rows is **Cardinality**

Column OR Attributes
Total # of column is **Degree**

Relational Integrity constraints

- Relational Integrity constraints is referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.
- There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:
 - Domain constraints
 - Key constraints
 - Referential integrity constraints

Domain Constraints

- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.
- Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.
- **Example:**
 - Create DOMAIN CustomerName CHECK (value not NULL)The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

Key constraints

- An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.
- Example:
 - In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Referential integrity constraints

- Referential integrity constraints is base on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Example:

we have 2 relations, Customer and Billing.


Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount \$300

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

InvoiceNo	CustomerID	Amount
1	1	\$100
2	1	\$200
3	2	\$150

Billing



Operations in Relational Model

- **SELECT (σ):** extracts tuples from a relation that satisfy a given restriction. Let R be a table that contains an attribute A. $\sigma_{A=a}(R) = \{t \in R \mid t(A) = a\}$ where t denotes a tuple of R and t(A) denotes the value of attribute A of tuple t.
- **PROJECT (π):** extracts specified attributes (columns) from a relation. Let R be a relation that contains an attribute X. $\pi_X(R) = \{t(X) \mid t \in R\}$, where t(X) denotes the value of attribute X of tuple t.
- **PRODUCT (\times):** builds the Cartesian product of two relations. Let R be a table with arity k1 and let S be a table with arity k2. $R \times S$ is the set of all k1 + k2-tuples whose first k1 components form a tuple in R and whose last k2 components form a tuple in S.
- **UNION (\cup):** builds the set-theoretic union of two tables. Given the tables R and S (both must have the same arity), the union $R \cup S$ is the set of tuples that are in R or S or both.
- **INTERSECT (\cap):** builds the set-theoretic intersection of two tables. Given the tables R and S, $R \cap S$ is the set of tuples that are in R and in S. We again require that R and S have the same arity.
- **DIFFERENCE ($-$ or \setminus):** builds the set difference of two tables. Let R and S again be two tables with the same arity. $R - S$ is the set of tuples in R but not in S.
- **JOIN (\Join):** connects two tables by their common attributes. Let R be a table with the attributes A,B and C and let S be a table with the attributes C,D and E. There is one attribute common to both relations, the attribute C. $R \Join S = \pi_{R.A,R.B,R.C,S.D,S.E}(\sigma_{R.C=S.C}(R \times S))$. What are we doing here? We first calculate the Cartesian product $R \times S$. Then we select those tuples whose values for the common attribute C are equal ($\sigma_{R.C=S.C}$). Now we have a table that contains the attribute C two times and we correct this by projecting out the duplicate column.



Best Practices for creating a Relational Model

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name
- No two rows can be identical
- The values of an attribute should be from the same domain

Advantages of using Relational model

- **Simplicity:** A relational data model is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
- **Query capability:** It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence:** The structure of a database can be changed without having to change any application.
- **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of using Relational model

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

Sample Queries

	complaint_id	product	company	zip_code	
0	469026	Credit card	Citibank	45247	469026-Credit card-Citibank-45247
1	469131	Credit card	Synchrony Financial	98548	469131-Credit card-Synchrony Financial-98548
2	479990	Credit card	Amex	78232	479990-Credit card-Amex-78232
3	475777	Credit card	Capital One	32226	475777-Credit card-Capital One-32226
4	469473	Credit card	Citibank	53066	469473-Credit card-Citibank-53066
5	470828	Credit card	Wells Fargo & Company	89108	470828-Credit card-Wells Fargo & Company-89108
6	470852	Credit card	Citibank	78249	470852-Credit card-Citibank-78249
7	479338	Credit card	JPMorgan Chase & Co.	19809	479338-Credit card-JPMorgan Chase & Co.-19809
8	480935	Credit card	Citibank	07018	480935-Credit card-Citibank-07018
9	469738	Credit card	Wells Fargo & Company	95409	469738-Credit card-Wells Fargo & Company-95409

Sample Queries

```
SELECT ccd.complaint_id, ccd.product, ccd.company, ccd.zip_code
FROM (SELECT complaint_id, product, company, zip_code
      FROM credit_card_complaints
      WHERE zip_code = '91701') ccd
LIMIT 10;
```


Sample Queries

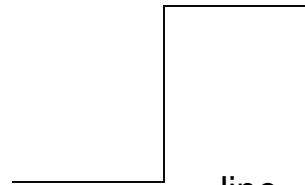
	complaint_id	product	company	zip_code
0	24857	Credit card	Barclays PLC	91701
1	33157	Credit card	Citibank	91701
2	12245	Credit card	Bank of America	91701
3	3151	Credit card	Barclays PLC	91701
4	352534	Credit card	Citibank	91701
5	1963836	Credit card	JPMorgan Chase & Co.	91701
6	2178015	Credit card	Discover	91701
7	2234754	Credit card	Discover	91701

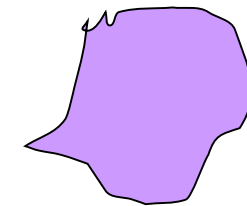
Topics we will learn 3/3

- Spatial data types
- Spatial Relationships
- Spatial Queries
- Spatial Indexing
- Vector Data Indexing and MBRs
- Spatial Indexing Methods
- SDBMS (Three-layer Structure)
- Geodatabase Model and queries
- POSTGIS Model and queries
- Spatial Database Applications

Spatial data types


point


line


polygon

- Point : 2 real numbers
- Line : sequence of points
- Region : area included inside n-points

Spatial Relationships

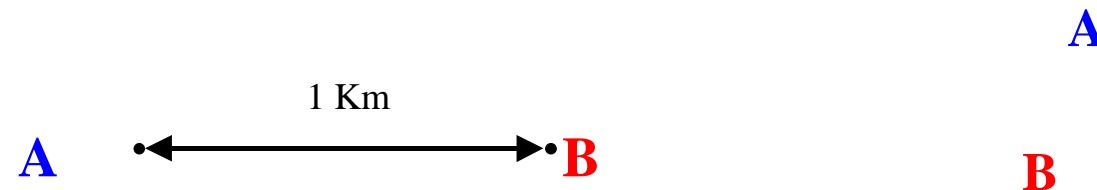
- Topological relationships:
 - adjacent, inside, disjoint, etc
- Direction relationships:
 - Above, below, north_of, etc
- Metric relationships:
 - "distance < 100"
- And operations to express the relationships

Spatial Relationships

- **Topological Relations:** containment, overlapping, etc. [Egenhofer et al. 1991]



- **Metric Relations:** distance between objects, etc. [Gold and Roos 1994]



- **Direction Relations:** north of, south of, etc. [Hernandez et al. 1990; Frank et al. 1991]

Spatial Queries

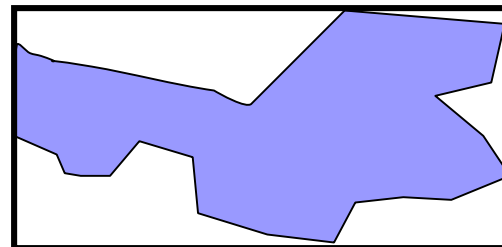
- Selection queries: "Find all objects inside query q ", inside- \rightarrow intersects, north
- Nearest Neighbor-queries: "Find the closest object to a query point q ", k -closest objects
- Spatial join queries: Two spatial relations $S1$ and $S2$, find all pairs: $\{x \text{ in } S1, y \text{ in } S2, \text{ and } x \text{ rel } y = \text{true}\}$, $\text{rel} = \text{intersect, inside, etc}$

Spatial Indexing

- Point Access Methods (PAMs) vs Spatial Access Methods (SAMs)
- PAM: index only point data
 - Hierarchical (tree-based) structures
 - Multidimensional Hashing
 - Space filling curve
- SAM: index both points and regions
 - Transformations
 - Overlapping regions
 - Clipping methods

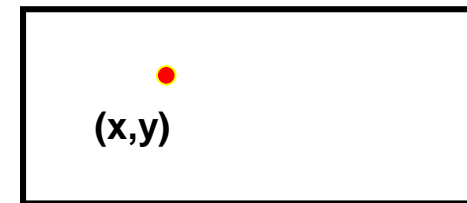
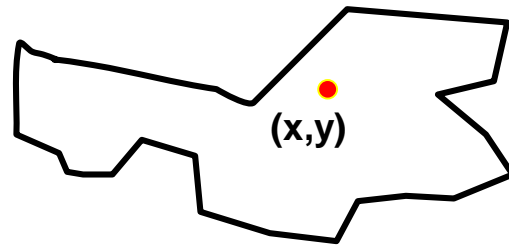
Vector Data Indexing

- Different indexing methods are used for **point**, **linear** and **polygonal** data
- In the case of collections of polygons, instead of indexing the object geometries themselves, whose shapes might be complex, we consider an approximation of the geometry and index it instead
- Most commonly used approximation: *minimum bounding rectangle* (MBR) also called *minimum bounding box* (MBB)



MBRs

- By using the MBR as the geometric key for building the spatial index, we save the cost of evaluating expensive geometric predicates during index traversal (as geometric tests against an MBR is constant)
- *Example:* point-in-polygon test



- In addition, the space required to store a rectangle is constant (2 points)

Spatial Indexing Methods

- The popular indexing technique those are in practice in industry standard RDBMS are:
 - Quadtree
 - Octree
 - UB-tree
 - R-tree
 - R+ tree
 - R* tree
 - Hilbert R-tree

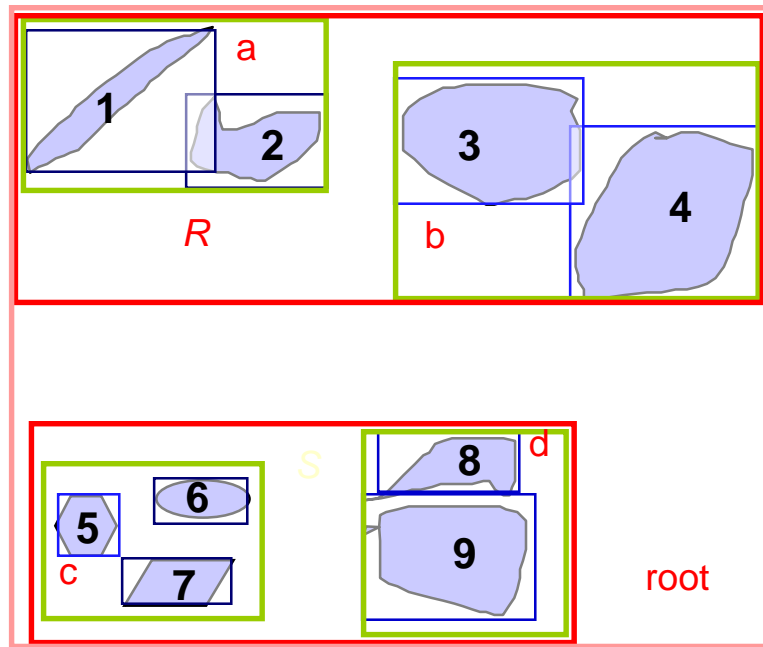
Two methods are most popular in SDBMS:

- R-trees
- Quadtrees

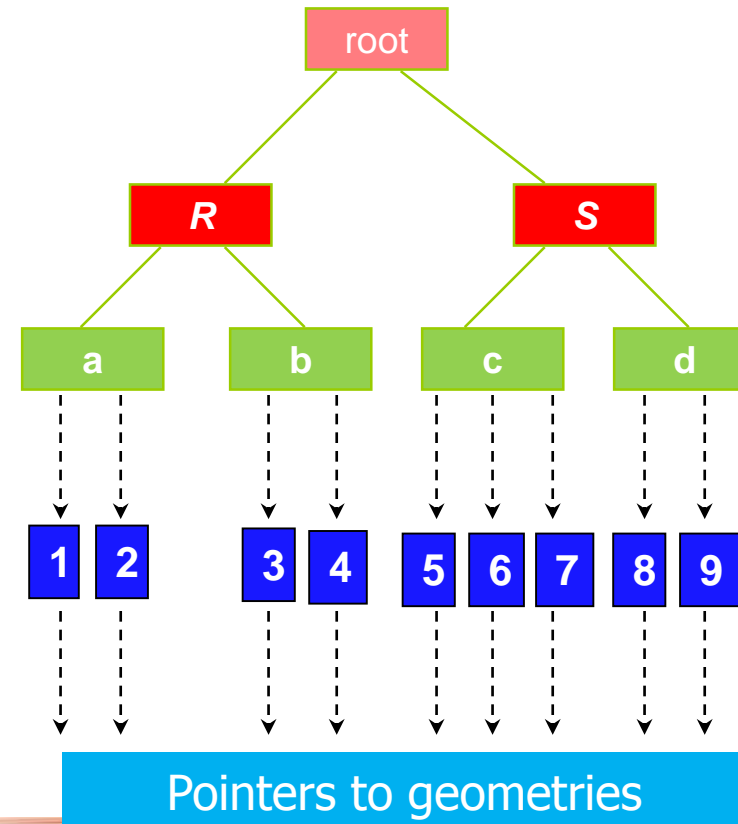
R-trees

- ❖ Based on MBRs (minimum bounding rectangles)
- ❖ Defined for indexing 2D objects (can be extended to higher dimensions but implemented only for 2D in Oracle Spatial)
- ❖ MBRs of geometric objects form the leaves of the index tree
- ❖ Multiple MBRs are grouped into larger rectangles (MBRs) to form intermediate nodes in the tree
- ❖ Repeat until one rectangle is left that contains everything

R-trees



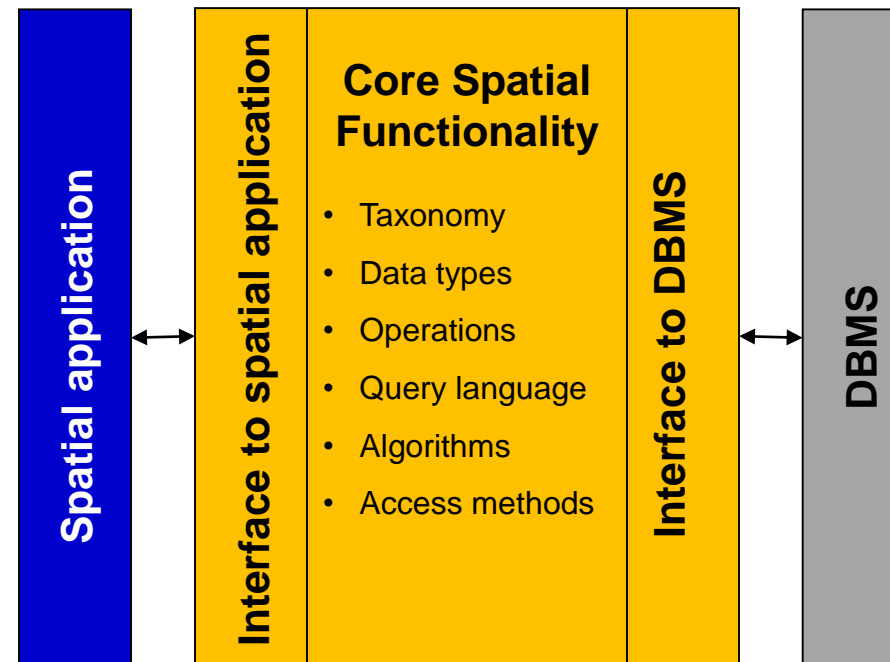
R-tree



SDBMS Three-layer Structure

- SDBMS works with a spatial application at the front end and a DBMS at the back end
- SDBMS has three layers:
 - Interface to spatial application
 - Core spatial functionality
 - Interface to DBMS

SDBMS Three-layer Structure (Cont.)



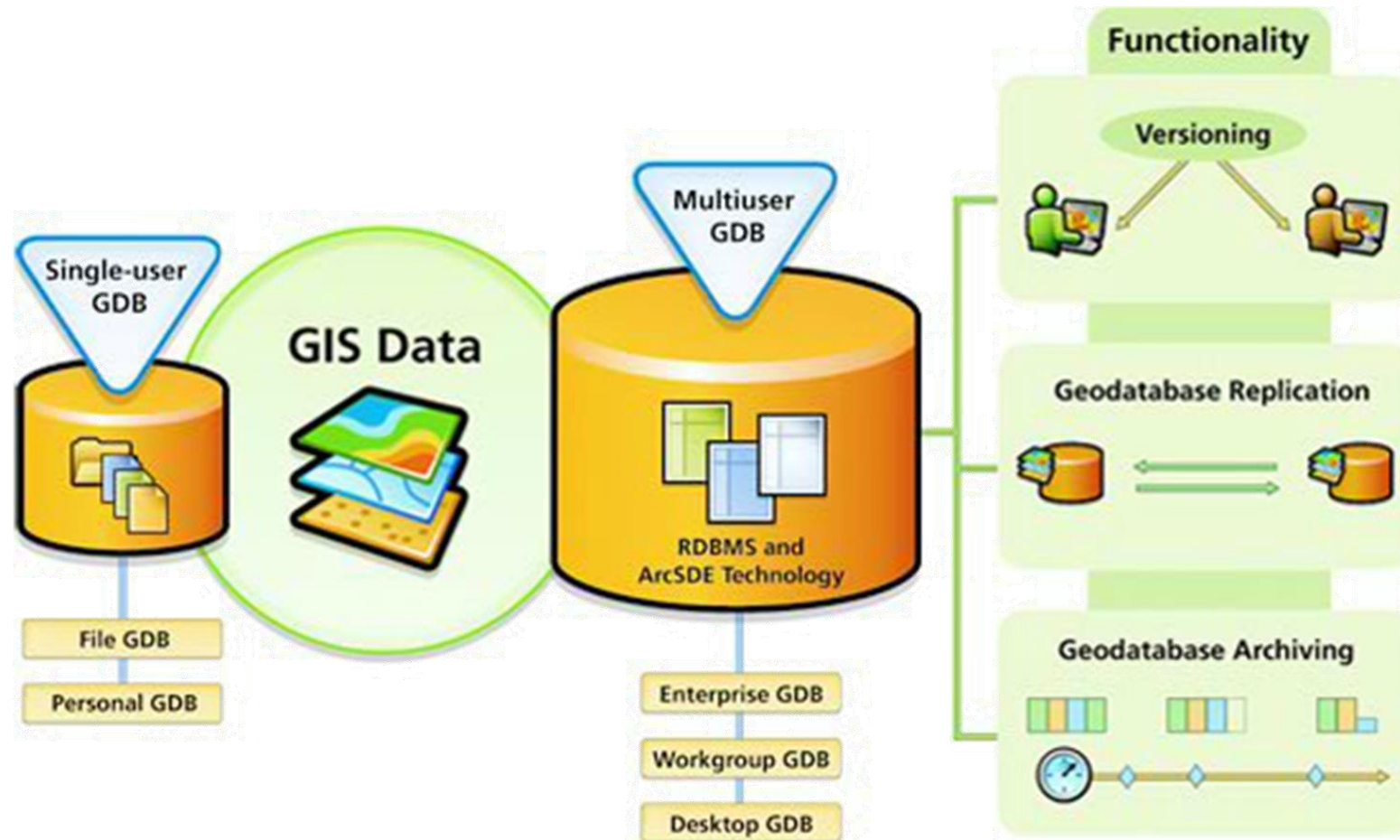
SDBMS

- The Organizing and management of geo-spatial data inside DBMS or RDBMS has emerged as an exciting and challenging area for GIS professionals during last decade which is known as Spatial Database Management System (SDBMS).

Geodatabase Model

The Geodatabase is the common data storage and management framework for ArcGIS. It combines "geo" (spatial data) with "database" (data repository) to create a central data repository for spatial data storage and management. It can be leveraged in desktop, server, or mobile environments and allows the users to store GIS data in a central location for easy access and management.

Geodatabase Model



Spatial Data Entity Creation

- Form an entity to hold county names, states, populations, and geographies

```
CREATE TABLE County(
    Name      varchar(30),
    State     varchar(30),
    Pop       Integer,
    Shape     Polygon);
```


Spatial Data Entity Creation (Cont.)

- Form an entity to hold river names, sources, lengths, and geographies

```
CREATE TABLE River(
    Name      varchar(30),
    Source     varchar(30),
    Distance   Integer,
    Shape      LineString);
```

Example Spatial Query

- Find all the counties that border on Contra Costa county

```
SELECT      C1.Name
```

```
FROM        County C1, County C2
```

```
WHERE Touch(C1.Shape, C2.Shape) = 1 AND C2.Name =  
      'Contra Costa';
```

Example Spatial Query (Cont.)

- Find all the counties through which the Merced river runs

```
SELECT      C.Name, R.Name
```

```
FROM        County C, River R
```

```
WHERE Intersect(C.Shape, R.Shape) = 1 AND R.Name =  
      'Merced';
```

POSTGIS Model

- PostGIS "**spatially enables**" the PostgreSQL server, allowing it to be used as a backend spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension.
- PostGIS follows the OpenGIS "Simple Features Specification for SQL" and has been certified as compliant with the "Types and Functions" profile.
- PostGIS development was started by Refrations Research as a project in open source spatial database technology.
- PostGIS is released under the GNU General Public License.
- PostGIS continues to be developed by a group of contributors led by a Project Steering Committee and new features continue to be added.

Types of queries - PostGIS (Cont.)

1. Distance(geometry, geometry) : number
2. Equals(geometry, geometry) : boolean
3. Disjoint(geometry, geometry) : boolean
4. Intersects(geometry, geometry) : boolean
5. Touches(geometry, geometry) : boolean
6. Crosses(geometry, geometry) : boolean

Types of queries - PostGIS (Cont.)

7. Overlaps(geometry, geometry) : boolean
8. Contains(geometry, geometry) : boolean
9. Intersects(geometry, geometry) : boolean
10. Length(geometry) : number
11. Area(geometry) : number
12. Centroid(geometry) : geometry

Spatial Database Applications

- GIS applications (maps):
 - Urban planning, route optimization, fire or pollution monitoring, utility networks, etc
- Other applications:
 - VLSI design, CAD/CAM, model of human brain, etc
- Traditional applications:
 - Multidimensional records

DISCLAIMER

The material for the presentation has been compiled from various sources such as books, tutorials (online and offline), lecture notes, several resources available on Internet. The information contained in this lecture/presentation is for general information and education purpose only. While we endeavor to keep the information up to date and correct, we make no representation of any kind about the completeness and accuracy of the material. The information shared through this presentation material should be used for educational purpose only.