Database Management Systems

Database:

A database is a structured collection of data that is organized and stored in a systematic way to allow for efficient retrieval, management, and manipulation of information. Databases can store a wide range of data types, from text and numbers to multimedia and complex data structures.

Database Users:

Database users are individuals or applications that interact with a database. They can be categorized into several roles, including:

1. **End Users:** These are the people who directly access the database to retrieve or input data. They typically use applications or query languages to interact with the database.
2. **Database Administrators (DBAs):** DBAs are responsible for managing and maintaining the database system. Their tasks include setting up user accounts, optimizing performance, and ensuring data integrity.
3. **Application Developers:** They create software applications that interact with the database, allowing users to perform specific tasks or functions.
4. **Data Analysts/Scientists:** These users extract and analyze data from the database to gain insights, make decisions, or perform statistical analyses.

Characteristics of Databases:

1. **Data Integrity:** Databases enforce rules to ensure the accuracy and consistency of data.
2. **Data Security:** Databases have mechanisms to control access to data and protect it from unauthorized users.
3. **Data Redundancy Reduction:** Databases minimize data duplication to save storage space and maintain consistency.
4. **Concurrent Access:** Databases allow multiple users to access and manipulate data simultaneously while ensuring data consistency.
5. **Data Recovery:** Databases provide backup and recovery mechanisms to restore data in case of failure.

6. **Query Language Support:** Databases offer query languages (e.g., SQL) for efficient data retrieval and manipulation.

Database Systems:

A database system is a software package that provides an interface for interacting with a database. It includes:

DBMS (Database Management System): The core component responsible for storing, managing, and retrieving data.

Database Application: Software that uses the DBMS to interact with the database.

Database Users: Individuals or applications that interact with the database.

Concepts and Architecture:

Three-tier Architecture: Common in web applications, with a presentation layer, application logic layer, and database layer.

Client-Server Architecture: Clients request data from a centralized server that manages the database.

Distributed Databases: Data is stored across multiple locations or servers.

ACID Properties: A set of properties (Atomicity, Consistency, Isolation, Durability) that ensure database transactions are reliable.

Database Models: Different ways to structure data, such as relational, NoSQL, hierarchical, and graph databases.

Data Models:

Relational Data Model: Organizes data into tables with rows and columns, and relationships are established using keys.

NoSQL Data Model: Diverse models (document, key-value, column-family, graph) that provide flexible data structures and scalability.

Hierarchical Data Model: Organizes data in a tree-like structure with parent-child relationships.

Graph Data Model: Represents data as nodes and edges to handle complex relationships.

Schemas & Instances:

Schema: A blueprint or structure that defines how data is organized in a database. It includes tables, columns, data types, constraints, and relationships.

Instance: An instance of a database is a specific set of data that conforms to the schema. It represents the actual data stored in the database.An instance is a specific set of data that conforms to a schema.

DBMS Architecture:

The architecture of a DBMS refers to the overall design of the system, including its components and how they interact with each other.

The three-level architecture of a DBMS consists of the external level (which defines how users view the data), the conceptual level (which defines the logical structure of the data), and the internal level (which defines how the data is physically stored on disk).

Data Independence:

Data independence is the ability to modify the conceptual schema without affecting the external schema or the physical schema.

Logical data independence refers to the ability to modify the conceptual schema without affecting the external schema.

Physical data independence refers to the ability to modify the physical schema without affecting the conceptual schema or the external schema.

Types of different database management system

Hierarchical Database Management Systems:

A hierarchical database system organizes data in a tree-like structure with each record having a parent record and zero or more child records. It was widely used in the 1960s

and 1970s for large-scale data processing applications. Hierarchical databases are simple and efficient, but less flexible than other types of databases. They are still used today in some specialized applications.

Network Database Management Systems:

A network database system is a type of database management system that organizes data in a more flexible structure than a hierarchical database, with each record having one or more parent records and one or more child records. It was developed in the 1960s and 1970s as an improvement over the hierarchical model. Network databases are more flexible than hierarchical databases, but they can be more complex to navigate. They are less commonly used today than relational databases, but are still used in some specialized applications.

Relational Database Management System

Relational DBMSs organize data in tables, with each table representing a set of related data and each row representing a single instance of that data.

Relational DBMSs use SQL (Structured Query Language) to manipulate data, including querying, inserting, updating, and deleting data.

Relational DBMSs provide a high level of data independence, making it easy to modify the structure of the database without affecting the applications that use it.

Relational DBMSs are widely used in business and other applications, due to their flexibility, scalability, and ease of use.

DBMS over File System

The adoption of a Database Management System (DBMS) resolved several key issues associated with traditional file systems:

1. Data Redundancy: In file systems, the same data could be stored in multiple places. DBMS centralized data storage to eliminate redundancy.

2. Data Inconsistency: File systems often had inconsistent data in different copies. DBMS ensures data consistency by maintaining a single version.

3. Data Access: File systems made data access difficult and insecure, especially for concurrent access. DBMS improved access efficiency and security.

4. Backup and Recovery: File systems lacked proper backup and recovery mechanisms, risking data loss. DBMS introduced systematic backup and recovery processes to protect data.

ER-Model

ER Diagram: An ER diagram is a model of a logical view of the database which is represented using the following components:

Entity: The entity is a real-world object, represented using a rectangular box.

Strong Entity: A strong entity set has a primary key and all the tuples of the set can be identified using that primary key

Weak entity: When an entity does not have sufficient attributes to form a primary key. Weak entities are associated with another strong entity set also known as identifying an entity. A weak entity's existence depends upon the existence of its identifying entity. The weak entity is represented using a double-lined or bold-lined rectangle.

Attribute: Attribute is the properties or characteristics of the real-world object. It is represented using an oval.

Key attribute: The attribute which determines each entity uniquely is known as the Key attribute. It is represented by an oval with an underlying line.

Composite Attribute: An attribute that is composed of many other attributes. E.g. address is an attribute it is formed of other attributes like state, district, city, street, etc. It is represented using an oval comprises of many other ovals.

Multivalued Attribute: An attribute that can have multiple values, like a mobile number. It is represented using a double-lined oval.

Derived attribute: An attribute that can be derived from other attributes. E.g. Age is an attribute that can be derived from another attribute Data of Birth. It is represented using a dashed oval.

Relationship: A relationship is an association between two or more entities. Entities are connected or related to each other and this relationship is represented using a diamond.

Some Important Terms

Cardinality of DBMS: Cardinality of relation expresses the maximum number of possible relationship occurrences for an entity participating in a relationship. Cardinality of a relationship can be defined as the number of times an entity of an entity set participates in a relationship set. Let's suppose a binary relationship R between two entity sets A and B. The relationship must have one of the following mapping cardinalities:

One-to-One: When one entity of A is related to at most one entity of B, and vice-versa.

One-to-Many: When one entity of A is related to one or more than one entity of B. Whereas B is associated with at most one entity in A.

Many-to-One: When one entity of B is related to one or more than one entity of A. Whereas A is associated with at most one entity in B.

Many-to-Many: Any number of entities of A is related to any number of entities of B, and vice-versa.

The most commonly asked question in ER diagram is the minimum number of tables required for a given ER diagram. Generally, the following criteria are used:

Cardinality	Minimum No. of tables
1:1 cardinality with partial participation of both entities	2
1:1 cardinality with a total participation of at least 1 entity	1
1:n cardinality	2
m:n cardinality	3

- If the relation is one-to-many or many-to-one then two or more relational tables can be combined.
- If the relation is many-to-many two tables cannot be combined.

- If the relation is one-to-one and there is total participation of one entity then that entity can be combined with a relational table.
- If there is total participation of both entities then one table can be obtained by combining one table and both entities of the relation.

Specialization: It is a top-down approach in which one entity is divided/specialized into two or more sub-entities based on its characteristics.

Generalization: It is a bottom-up approach in which common properties of two or more subentities are combined/generalized to form one entity. It is exactly the reverse of Specialization. In this, two or lower level entities are generalized to one higher level entity.

Aggregation: Aggregation is an abstraction process through which relationships are represented as higher-level entity sets.

Participation Constraint: It specifies the maximum or a minimum number of relationship instances in which any entity can participate. In simple words, participation means how an entity is linked to a relationship.

- Total Participation: Each entity of an entity set participates in at least one relationship.
- Partial Participation: Some entities of the entity set may not participate in any relationship.

Database Design

Database design Goals: The prime goal of designing a database is:

- To have zero redundancy in the system
- Loss-less join
- Dependency preservation
- Overcome all the shortcomings of conventional file system

According to E.F. Codd, "All the records of the table must be unique".

Keys of a relation: There are various types of keys in a relation which are:

- **Candidate Key:** The minimal set of attributes that can determine a tuple uniquely. There can be more than 1 candidate key of a relation and its proper subset can determine tuple uniquely and it can be NULL.
- **Super Key:** The set of attributes that can determine a tuple uniquely. A candidate key is always a super key but vice versa is not true.
- **Primary Key and Alternate Key**: Among various candidate keys, one key is taken as the primary key and others are alternate keys.
- **Foreign Key:** Foreign Key is a set of attributes in a table that is used to refer to the primary key or alternative key of the same or another table.

Functional Dependency: It is a constraint that specifies the association/ relationship between a set of attributes. In functional dependency, one set can accurately determine the value of another set. It is represented as A&#x2192B, where set A can determine the values of set B correctly. The A is known as the Determinant, and B is known as the Dependent.

Functional dependencies are further categorized into two types:

- Trival Functional Dependency: In functional dependency, if B is a subset of A, then such dependency is known as trivial functional dependency.
- Non-Trivial Functional Dependency: In functional dependency, if B is not a subset of A, then such dependency is known as non-trivial functional dependency.

Armstrong's Axioms: It is a statement that is always considered true and used as a starting point for further arguments. Armstrong axiom is used to generate a closure set in a relational database.

Armstrong Axiom	
Primary Rules	**Secondary Rules**
Reflexive: If Y⊆ X then X→ Y	**Union:** If X→Y, and Y→Z, then X→YZ
Transitivity: If X→Y and Y→Z, then X→Z	**Decomposition:** If X→YZ, then X→Y and X→Z
Augumentation: If X→Y , then XZ→YZ	**Composition:** If X→Y and Z→W, then XZ→YW
	Pseudo Transitivity: If X→Y and WY→Z, then WX→Z

Armstrong Axiom

Attribute Closure (X+): This indicates that all attributes in the set X are functionally determined by X. For instance, in a relation R(ABCD), A+ would be {ABCD}, as A determines B, B determines C, and C determines D.

Prime Attribute: A prime attribute is one that is part of at least one candidate key. For example, in R(ABCD), A and B are prime attributes because they are part of the candidate key.

Non-prime Attribute: A non-prime attribute is not part of any candidate key. In R(ABCD), C and D are non-prime attributes.

Equivalence Sets of Functional Dependency: Two sets of functional dependencies A and B are considered equivalent if A+ = B+, meaning that all the dependencies in A can be inferred from B, and vice versa.

Minimal Set of Functional Dependency: A set of functional dependencies is minimal if it logically implies all dependencies and if no dependency in the set contains an extraneous attribute.

Normalization and Anomalies:

Normalization: This is the process of organizing a relational database to eliminate anomalies, improve integrity, and maintainability.Normalization is used to eliminate the following anomalies:

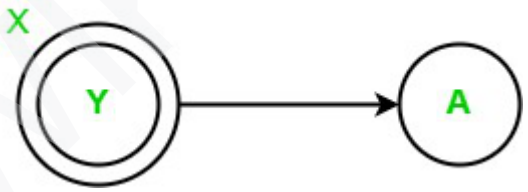
- Insertion anomaly
- Deletion anomaly
- Updation Anomaly
- Join anomaly

First Normal Form (1NF): A relation is in 1NF if it doesn't contain multi-valued or composite attributes, ensuring there's no data redundancy.

Second Normal Form (2NF): A relation is in 2NF if it's in 1NF and has no partial dependencies. Partial dependency occurs when a proper subset of the candidate key determines non-prime attributes.

Partial Dependency: A dependency is called partial dependency if any proper subset of candidate key determines non-prime (which are not part of candidate key) attribute.

Let R be the relational schema and X, Y, A is the set of attributes. Suppose X is any candidate key, Y is a proper subset of candidate key, and A is a Non-prime attribute.



Y will be partial dependency iff, Y is a proper subset of candidate key, and A is a non-prime attribute.

Full Functional Dependency: If A and B are an attribute set of a relation, B is fully functional dependent on A, if B is functionally dependent on A but not on any proper subset of A.

Third Normal Form (3NF): A relation is in 3NF if it's in 2NF and has no transitive dependencies.

Boyce-Codd Normal Form (BCNF): A relation is in BCNF if the left-hand side of every functional dependency is a superkey.

Design Goal	1NF	2NF	3NF	BCNF
Zero Redundancy	High redundancy	Less than 1NF	Less than 2NF	No redundancy
Loss-less decomposition	Always	Always	Always	Always
Dependency preservation	Always	Always	Always	Sometimes Not possible

Properties of Decomposition:

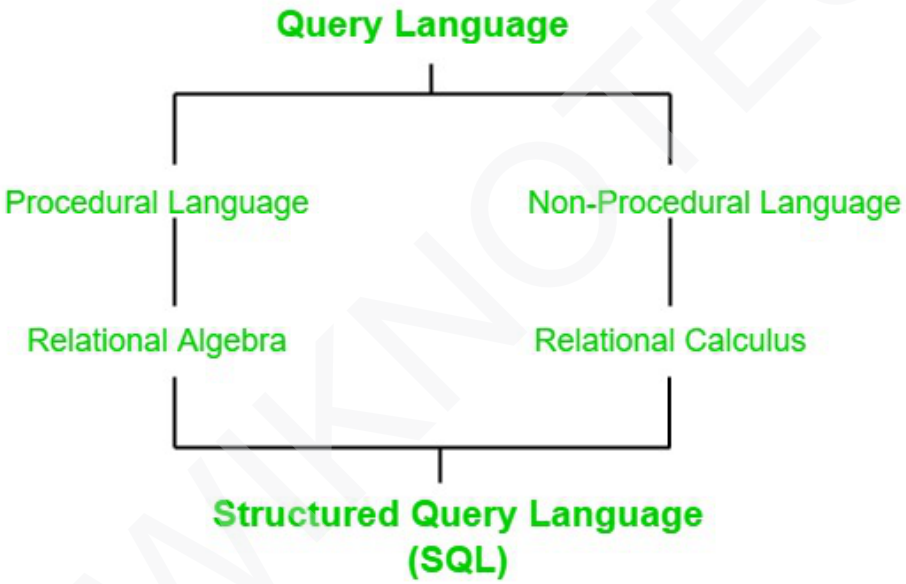
Loss-less Join Decomposition: Decomposition should not create new tuples, which is represented as R ≡ R1 ∗ R2.

Dependency Preserving Decomposition: Decomposition should not lose any dependencies. If F1 U F2 U ... U Fn ≡ F, the decomposition is considered dependency-preserving.

Data Retrieval (SQL, RA):

- **DDL (Data Definition Language):** DDL includes commands like CREATE, ALTER, DROP, COMMENT, and TRUNCATE for defining the database structure.
- **DML (Data Manipulation Language):** It involves commands like SELECT, INSERT, DELETE, UPDATE for data manipulation.
- **DCL (Data Control Language):** DCL manages access rights using commands like GRANT and REVOKE.

Query Language: This is the language used to retrieve data from a database, with SQL being a common example.



Relational Algebra:

Basic Operators: Basic operators include σ (Selection), ∏ (Projection), × (Cross Product), ∪ (Union), and − (Difference).

Extended Operators: Extended operators include ∩ (Intersection), ⋈ (Join), ⋈θ (Equi Join), ⋈N (Natural Join), ⟕ (Left Outer Join), ⟖ (Right Outer Join), ⟗ (Full Outer Join), and / (Division Operator).

Design Goals in Normalization:

The aim of normalization is to reduce redundancy and avoid anomalies, such as insertion, deletion, updation, and join anomalies. Each level of normalization (1NF, 2NF, 3NF, BCNF) brings the database closer to an ideal state free from redundancy and anomalies.

Note: The relational model is a theoretical framework, while the Relational Database Management System (RDBMS) is its practical implementation. Relational algebra is a procedural language used for working with relational databases.

Relational Calculus:

Relational Calculus is a non-procedural query language that specifies what to do, but not how to do it. It comes in two types:

Tuple Relational Calculus: It is based on specifying the number of tuple variables, with each variable ranging over a particular database relation. It takes the form {t| cond(t)}, where t is the tuple variable, and cond(t) is a conditional expression. The result is the set of tuples satisfying cond(t).

Domain Relational Calculus: In this type, variables range over single values from domains of attributes. It is structured as {x1, x2, ..., xn | cond (x1, x2, ..., xn, xn+1, xn+2, ..., xn+m)}, where x1, x2, ..., xn, xn+1, xn+2, ..., xn+m are domain variables, and cond is a condition.

SQL (Structured Query Language):

SQL is a powerful language for accessing and modifying databases. It can execute various operations such as queries, data retrieval, record insertion, record updating, record deletion, database creation, table creation, view creation, and setting permissions on tables, procedures, or views.

SQL Commands and Their Meanings:

- SELECT: It selects columns from a relation or set of relations and defines what data to retrieve.
- FROM: Specifies the tables or views used in the SELECT or WHERE statements.
- WHERE: Defines which records are included in the query based on conditional operators.
- EXISTS: Checks if the result of a correlated nested query is empty.
- GROUP BY: Groups tuples based on specific attributes, allowing operations like counting the number of students in each department.
- ORDER BY: Sorts the retrieved data in ascending or descending order according to one or more columns.
- Aggregate functions: These functions calculate aggregated values of an attribute, often used with GROUP BY (e.g., count, sum, min, max).
- Nested Queries: When one query is part of another query.
- UPDATE: Used to modify records in a table.
- DELETE: Used to remove rows from a table.
- LIKE: Operates with the WHERE clause to search for a specified pattern in a column.
- IN: Specifies multiple values in the WHERE clause.
- BETWEEN: Selects values within a specified range.
- Aliases: Temporarily rename a table or column heading.
- HAVING: Used with aggregate functions to filter the results.

File Structure:

File organization defines the logical relationship between records and how they are mapped into disk blocks. A database consists of files, each containing records with various fields. The blocking factor represents the average number of records per block.

Two common strategies for storing files of records are:

Spanned Strategy: This allows partial storage of a record in a block and is suitable for variable-length records, but it can increase block access time.

Unspanned Strategy: Data cannot be stored partially; each block is fully occupied. This can lead to internal fragmentation and wasted memory but reduces block access time. It's suitable for fixed-length records.

File organizations include:

- Sequential File Organization
- Heap File Organization
- Hash File Organization
- B+ Tree File Organization
- Clustered File Organization

Each of these has its own characteristics and advantages for specific use cases.

Transaction and Concurrency Control:

A transaction is a logical unit of work that is atomic, consistent, isolated, and durable (ACID properties). Transactions can be managed using different methods:
Serializability: Ensures that a schedule of transactions is equivalent to a serial schedule.
Conflict Serializability: Achieved by swapping non-conflicting operations in a schedule.
View Serializability: Achieved when a schedule is view-equivalent to a serial schedule.
To handle concurrent transactions and maintain consistency, locking protocols are used. These protocols restrict the set of possible schedules and can be binary (locked/unlocked) or shared/exclusive. Two-phase locking ensures that transactions go through growing and shrinking phases, and each transaction locks data items before using them.
Concurrency control with locks helps achieve isolation and consistency by managing how transactions access data concurrently.

Time Stamp Ordering Protocols:
Time stamp protocols assign unique timestamps to transactions and establish an order for transaction submission. The Thomas Write Rule, in the context of time-stamping, helps maintain consistency by allowing or preventing write operations based on timestamps.

These protocols are used to manage transaction concurrency, ensuring that transactions are processed in an orderly and coordinated manner.