# DATABASE FUNDAMENTALS

**RDBMS** (Relational Database Management System)

Type of database system Stores and manages data in a structured and organized way. Follows the relational model emphasizes relationships between different pieces of data.

Key concepts of RDBMS:

Tables:

Data is stored in tables, Each table represents a specific entity (e.g., customers, products, orders).

Rows and Columns:

*Tables consist of rows (records) and columns (attributes).

*Each row represents a single instance of the entity (e.g., a specific customer record).

* Columns define the characteristics of that entity (e.g., customer name, address, email).

Keys:

*Primary key - A unique identifier for each record. eliminates duplicate data entries. cannot be null

(e.g. emp_Id in emp table )

*Foreign keys - reference primary keys in other tables establishing relationships between them.

(e.g. deptId in emp table )

*Unique keys - unique and can be null

*Candidate key - combination of one or more attributes that can uniquely identify a tuple (row) in a relation (table).

Relationships:

Tables are not isolated data silos. RDBMS allows you to define relationships between tables. This

enables you to efficiently retrieve and manipulate data across multiple tables based on these

connections.

**SQL (Structured Query Language):**

Standard language for interacting with relational databases. Allows you to perform various operations

like query data insert new records ,update existing data, delete data.

CRUD: create, read, update, delete

Benefits of RDBMS:

>Structured Data Organization: Data is organized in a clear and well-defined manner easier to

understand, manage, and query.

>Data Integrity: ACID properties ensure data consistency and accuracy.

>Data Relationships: Ability to define relationships between tables simplifies complex data retrieval and manipulation.

>SQL Support: SQL widely used and standardized language for querying and managing data.Modify

the schema .

DDL- data definition language : create, drop, truncate

DML - data manipulation language : insert, update, delete

DCL - data control language : grant and revoke

DQL - data query language : select

TCL- transaction control language

>Scalability: RDBMS can scale to accommodate large datasets and high volumes of transactions.But cannot scale like no-sql.

**Phantom Reads**

In database systems a Phantom Read type of concurrency anomaly that can occur when multiple transactions are executed concurrently.

Phantom reads can lead to inconsistent results and data integrity issues, especially in applications that rely on accurate counts or summaries of data.

How to prevent Phantom Reads:

Higher Isolation Levels: Using higher isolation levels like Serializable or Repeatable Read can prevent phantom reads. These levels typically employ techniques like locking to prevent other transactions from modifying data that is being accessed by a current transaction.

Serializable Vs Repeatable read

Repeatable Read

Prevents: Non-repeatable reads (where a single transaction reads the same row multiple times, and gets different values each time).

Allows: Phantom reads (where new rows meeting the same criteria as a previous read appear in subsequent reads within the same transaction).

How it works: Typically uses row-level locks to ensure that no other transaction can modify a row that has been read by the current transaction.

Serializable

Prevents: Both non-repeatable reads and phantom reads.

How it works: Employs stricter locking mechanisms, often involving predicate locks or snapshot isolation. This ensures that the execution of concurrent transactions is equivalent to executing them serially (one after the other), preventing any anomalies.

Eventual Consistency

Eventual Consistency consistency model used in distributed systems, particularly in those that prioritize availability and fault tolerance over immediate consistency.

ACID PROPERTIES

ACID stands for

Atomicity, Consistency, Isolation, and Durability.

>Atomicity: Ensures transaction is treated as a single unit

>Consistency: Data is said to be consistent if it remains the same through the life of a transaction. Guarantees transaction maintains the data according to pre-defined rules (constraints)

>Isolation: Ensures concurrent transactions are isolated from each other, prevents conflicts and maintaining data integrity.

> Durability: Guarantees that once a transaction is committed (finalized), the changes are permanently saved to the database.

**CAPS THEOREM**

The CAP Theorem (also known as Brewer's Theorem) is a principle that applies to distributed databases. It states that a distributed database system can achieve at most two of the following three goals simultaneously:

1. Consistency (C): Every read operation will return the most recent write. All nodes in the system have the same data at the same time.

2. Availability (A): Every request (read or write) will receive a response, even if some of the database nodes are down. The system is always available for operations.

3. Partition Tolerance (P): The system can continue to operate correctly even if network partitions occur, meaning some parts of the system can't communicate with others.

Sharding in RDBMS

Sharding is a method of horizontal scaling where data is distributed across multiple servers (called shards) to improve performance and manage large datasets. Two Types of Sharding:

1. Horizontal Sharding (Data Sharding): Data is split across multiple databases or servers based on some criteria.

2. Vertical Sharding (Database Sharding): Different tables or ty pes of data are stored on different servers. Each shard holds a subset of the database schema.

Normalization

Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies. The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Basic SQL Commands:

Here's a glimpse into some fundamental SQL commands to get you started:

SELECT: used to retrieve data from one or more tables. can specify columns (attributes) filter the results using WHERE clauses.

INSERT: insert new rows of data into a table. specify table name values for each column in the new row.

UPDATE: modify existing data in a table. can update specific columns based on a WHERE clause to target the rows you want to modify.

DELETE: This command removes rows from a table. use a WHERE clause to delete specific rows based on criteria.

CREATE TABLE: create new tables within the database defining the structure with column names and data types

Indexes in Databases

• An index in a database is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional space and increased maintenance during data modifications (INSERT, UPDATE, DELETE).

• Indexes are primarily used to speed up the search (querying) operations but can also impact the performance of write operations.

CREATE INDEX idx_customerid ON Customer (CustomerID);

//Create database

CREATE DATABASE database_name;

//use database

Show DATABASE;

//create table

CREATE TABLE people ( id BIGINT PRIMARY KEY, name varchar(500), email varchar(500), first_name varchar(500), last_name varchar(500), state varchar(200), birthday datetime );

//insert into table

INSERT INTO people (id, name, email, first_name, last_name, state, birthday) VALUES (1,'John Doe', 'john.doe@example.com', 'John', 'Doe', 'California', '1990-01-01')

//view table

Select * from people;

//DQL – Data Query Language

```
SELECT column1, column2, ... FROM table_name WHERE condition;

// Data Definition Language (DDL)

CREATE DATABASE database_name;

CREATE TABLE table_name ( column1 datatype, column2 datatype, ... );

ALTER TABLE table_name ADD column_name datatype;

DROP DATABASE database_name;

DROP TABLE table_name;

// Data Manipulation Language (DML)

INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);

UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

DELETE FROM table_name WHERE condition;

// Data Control Language (DCL)

GRANT SELECT, INSERT ON Employees TO user1;

REVOKE SELECT ON Employees FROM user1;

// Transaction Control Language (TCL)

COMMIT;

ROLLBACK;

SAVEPOINT savepoint_name;

SET TRANSACTION ISOLATION LEVEL level;
```