**What is Database?**

Database is an electronic place/system where data is stored in a way that it can be easily accessed, managed, and updated.

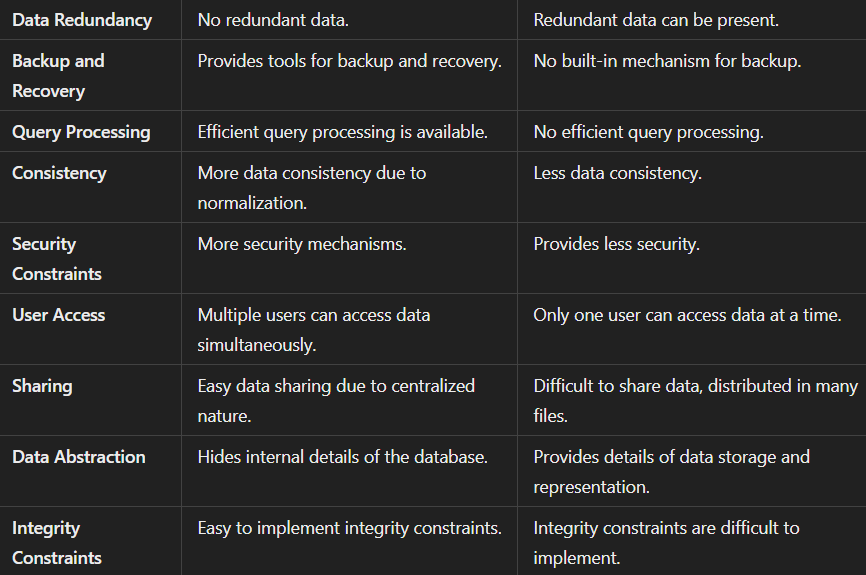
What is DBMS?

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Advantages of DBMS

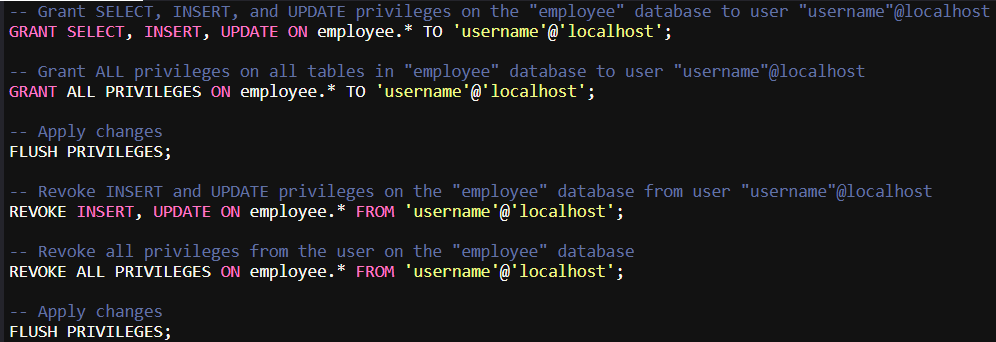
* **Data Redundancy Control**: DBMS minimizes data duplication by centralizing data management, leading to a more efficient storage system.
* **Data Integrity**: It ensures the accuracy and consistency of data through integrity constraints and validation rules.
* **Data Security**: DBMS provides access controls and user permissions to protect sensitive information from unauthorized access.
* **Data Consistency**: Changes made to data are reflected across the database, maintaining uniformity.
* **Data Backup and Recovery**: DBMS facilitates data backup and recovery options to prevent data loss due to failures.
* **Multi-user Support**: It allows multiple users to access and manipulate the database simultaneously while maintaining data integrity.

DBMS vs File Systems



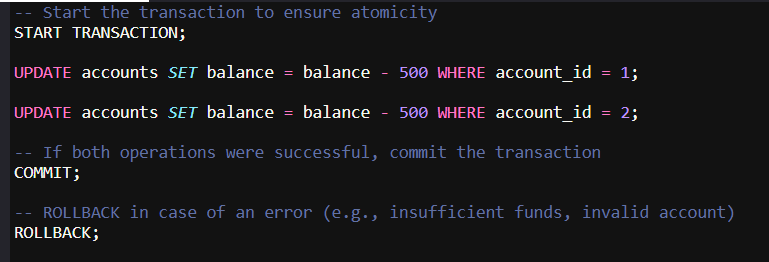
What are the different languages used in DBMS?

* **(DDL):** CREATE, ALTER, DROP, TRUNCATE, and RENAME.
* **(DML):**  SELECT, UPDATE, INSERT, DELETE, etc.
* **(DCL):** GRANT and REVOKE.
* **(TCL):**  COMMIT, ROLLBACK, and SAVEPOINT.



MySQL caches privilege information for performance, so changes to user privileges may not be recognized immediately unless this command is issued. Running FLUSH PRIVILEGES; tells MySQL to clear its cache and reload privilege data, applying recent changes instantly.

employee.\* grants or revokes permissions on every table within the employee database for the specified user.



**Commit**: A commit operation saves all changes made during the transaction to the database. Once a commit is executed, the changes become permanent.

**Rollback**: A rollback operation undoes all changes made during the current transaction. This is useful when an error occurs, and you want to revert the database to its previous state.

### Types of keys? (7 types)

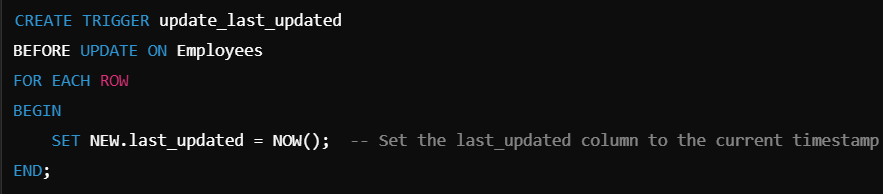
1. **Super Key (SK)**: Any combination of attributes that can uniquely identify each row in a table.
2. **Candidate Key (CK)**: A minimal subset of super keys that can uniquely identify each record. It has no unnecessary attributes.
3. **Primary Key (PK)**: The selected candidate key that uniquely identifies each record in a table. It does not allow null values.
4. **Foreign Key (FK)**: An attribute or set of attributes in one table that refers to the primary key of another table, creating a relationship between tables.
5. **Composite Key**: A primary key that consists of two or more attributes combined to uniquely identify a record.
6. **Alternate Key**: Any candidate key not chosen as the primary key.
7. **Unique Key**: Ensures uniqueness of values in a column but can allow null values. It’s similar to a primary key but with the flexibility to include nulls.

The following constraints are commonly used in SQL:

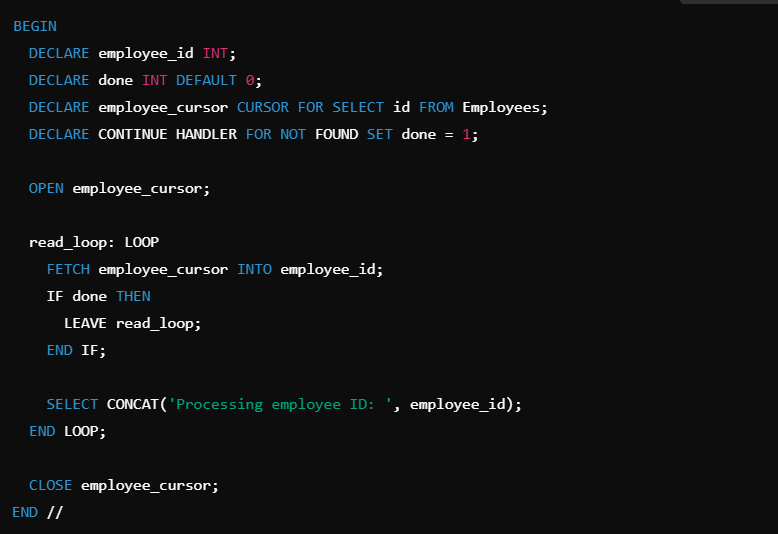
* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly

**Trigger:** A trigger is a set of commands that automatically execute in response to specific events on a table.

1. Data Definition Language (DDL) Triggers
2. Data Manipulation Language (DML) Triggers
3. Logon Triggers: A LOGON trigger in SQL Server can be created to execute a specific action whenever a user session is established.



**Cursor**: A cursor in SQL is a database object that allows us to retrieve each row at a time and manipulate its data. A cursor is nothing more than a pointer to a row. The SQL cursor's purpose is to update the data row by row, change it, or perform calculations that are not possible when we retrieve all records at once.



Declare Cursor: Use *DECLARE cursor\_name CURSOR FOR select\_statement;* to define the cursor with a SELECT query.

Open Cursor: Use OPEN cursor\_name; to initialize the cursor and retrieve data from the SELECT statement.

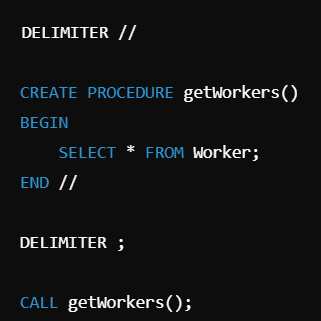
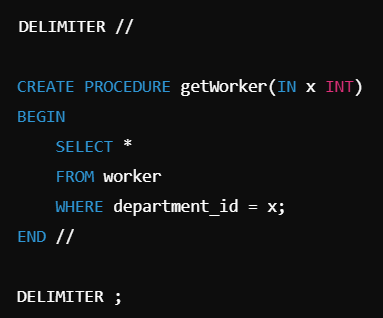
Fetch Cursor: Use FETCH NEXT FROM cursor INTO variable\_list; to retrieve rows one by one for data manipulation.

Close Cursor: Use CLOSE cursor\_name; to release the cursor after finishing data manipulation.

Deallocate Cursor: Use DEALLOCATE cursor\_name; to free resources and remove the cursor definition.

A **stored procedure** is a precompiled collection of one or more SQL statements stored in the database. Stored procedures can accept parameters, execute complex operations, and return results. They are useful for encapsulating business logic, improving performance, and enhancing security.

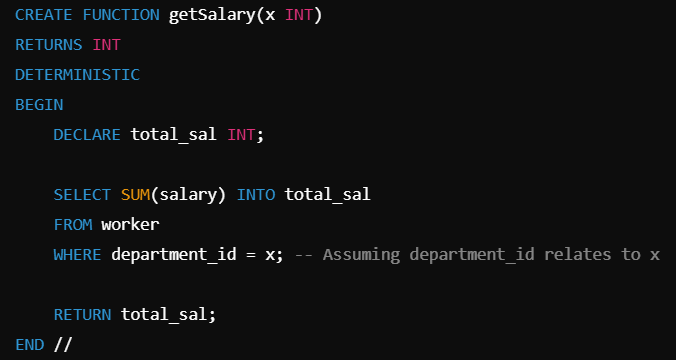
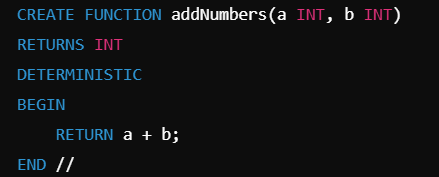
DROP PROCEDURE IF EXISTS procedure\_name;

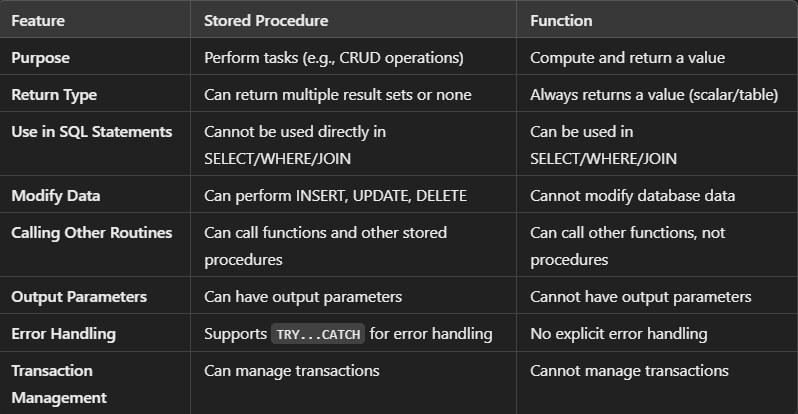
 

A **function** in SQL is a stored program that returns a single value. Functions can take parameters and are typically used to perform calculations or return a specific piece of information. Unlike stored procedures, functions can be called within SQL statements.

DROP FUNCTION IF EXISTS function\_name;

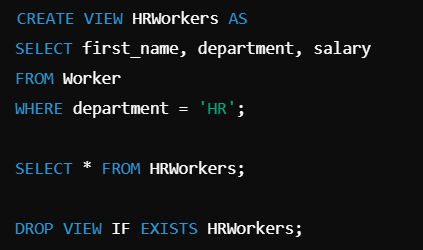
SELECT addNumbers(5, 7);



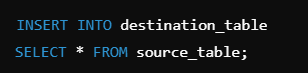


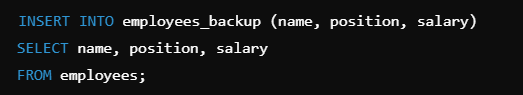
A **database schema** is the blueprint that defines the structure, organization, and relationships of data within a database.

**View** schema describes the database part that a particular user group is interested and hides the remaining database. a database contains several schemas that sometimes called as subschema, views also provide a security mechanism to prevent users from accessing certain parts of DB.



To copy the entire data from one table to another table in MySQL, you can use the INSERT INTO statement with a SELECT query. Here's the syntax:





What is the purpose of the SQL TRUNCATE statement?

The TRUNCATE statement is used to remove all rows from a table quickly. Unlike the DELETE statement, which removes rows one at a time and logs each deletion, TRUNCATE deallocates the entire data storage and is faster. However, it cannot be rolled back, and it does not trigger any DELETE triggers.

TRUNCATE TABLE Categories;

SQL (Structured Query Language) is a standardized programming language used to manage, manipulate, and query relational databases.

Relational Database Management System (RDBMS) is a type of database management system that stores data in a structured format using rows and columns. It uses a schema to define the tables and their relationships, allowing for efficient data retrieval and manipulation.

**MySQL** is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage, manipulate, and retrieve data.

Difference between SQL and MySQL: SQL (Structured Query Language) is the standard language used to query and manage databases, while MySQL is an RDBMS that uses SQL to perform database operations.

CHAR is a fixed-length data type that reserves a specified amount of space, while VARCHAR is a variable-length data type that only uses space based on the actual string length.

**Types of Functional Dependency (FD):**

1. **Trivial FD:** A → B is trivial if B ⊆ A (e.g., A→A, B→B).
2. **Non-trivial FD:** A → B is non-trivial if B is not a subset of A (A ∩ B = NULL).
3. **Multivalued Dependency**
4. **Transitive Dependency**

**Rules of FD (Armstrong’s Axioms):**

1. **Reflexive:** If B ⊆ A, then A → B.
2. **Augmentation:** If A → B, then AX → BX (adding attributes doesn’t change dependency).
3. **Transitivity:** If A → B and B → C, then A → C.

**What is Redundancy?**

**Redundancy** in the unnecessary duplication of data. It occurs when the same piece of data is stored in multiple places within a database. This can lead to various issues, such as increased storage costs, data inconsistency, and complications in data management.

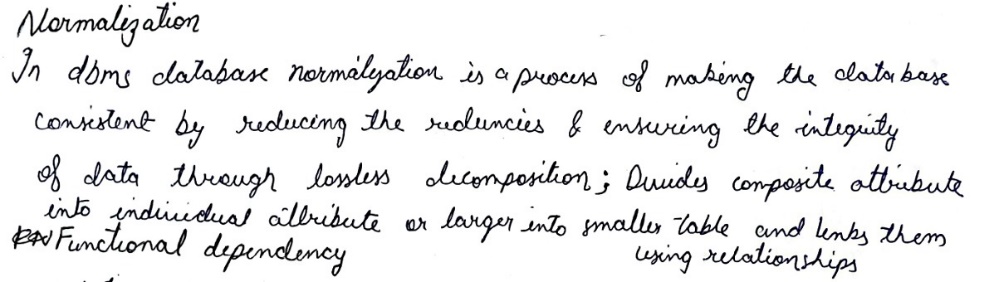
For example, if a customer's contact information is stored in multiple tables across a database, that contact information might get updated in one place but not in others, causing inconsistencies.

Problems: Insertion, deletion, updating anomilies.

How to Remove Redundancy:

* **Normalization**: Organize data into smaller, related tables to ensure that each piece of data is stored only once, following normal forms (1NF, 2NF, 3NF) to eliminate unnecessary repetition.
* **Use Foreign Keys**: Reference related data from other tables using foreign keys instead of duplicating the same information across multiple tables.
* **Apply Data Integrity Constraints**: Implement primary keys and unique constraints to enforce uniqueness and prevent duplicate records from being inserted into the database.

**Normalization:**



* Evaluates schemas based on functional dependencies.
* Minimizes redundancy and eliminates anomalies (Insertion, Update, Deletion).
* Divides composite attributes or large tables into smaller linked tables.

**Normal Forms:**

* **1NF:** Atomic values, no multi-valued attributes.
* **2NF:** In 1NF, no partial dependencies.
* **3NF:** In 2NF, no transitive dependencies.
* **BCNF:** In 3NF, each determinant is a super key. A -> B, A must be a super key.
* 4NF: has no multi-valued dependencies.
* 5NF: 4NF and supports lossless decomposition, handling join dependencies.

**Denormalization** is the process of intentionally introducing redundancy into a database to improve read operations and reduce the need for complex joins, trading off some data integrity for faster query execution in performance-critical scenarios.

**Transaction:**

* A unit of work against the DB in a specific sequence.
* Contains one or more SQL statements; either all succeed, or all are undone.

**ACID Properties:**

1. **Atomicity:** All or nothing; either all changes are applied or none.
2. **Consistency:** Data integrity is maintained before and after the transaction.
3. **Isolation:** Transactions execute independently without interference.
4. **Durability:** Changes persist even after system failures.

**Indexes** are database objects that improve the speed of data retrieval operations on a database table. An index is a data structure that provides a quick lookup capability for the database. While they can greatly enhance query performance, they also require additional storage space and can slow down data modification operations (like INSERT, UPDATE, DELETE) because the index must be updated as well.

CREATE INDEX idx\_employee\_name ON Employees(name);

**Types of Indexes:**

* **Primary Index:** An ordered file with fixed-length records containing the primary key and a pointer to the corresponding data block, requiring an average of log⁡2Bi+1\log\_2 B\_i + 1log2​Bi​+1 block accesses, where BiB\_iBi​ is the number of index blocks.
* **Clustering Index:** An index created on a data file with records physically ordered based on a non-key field, known as the clustering field.
* **Secondary Index:** An additional index providing alternative access to a file that already has a primary access method.

**Ordered Indexing Types**

1. **Dense Index:** Records for every search key; faster but more storage.
2. **Sparse Index:** Some keys are missing; uses pointers, sequential search if not found.
3. **Multilevel Index:** Breaks large indices into smaller ones for efficient memory use.

NoSQL databases (aka "not only SQL")

* types are document, key-value, wide-column, and graph.
* They are schema free.
* Data structures used are not tabular, they are more flexible, has the ability to adjust dynamically.
* Most of the NoSQL are open sources and has the capability of horizontal scaling.

**Advantages of NoSQL Databases**: Flexible Schema, Horizontal Scaling, High Availability, Fast Operations, Caching, Cloud-Friendly.

What is NoSQL?:

NoSQL databases are non-relational databases designed to handle large volumes of structured, semi-structured, and unstructured data. They offer flexibility, scalability, and high availability, making them suitable for big data and real-time web applications.

Types of NoSQL Databases:

* **Document Stores** (e.g., MongoDB): Store data as documents (JSON-like format) that can be queried easily.
* **Key-Value Stores** (e.g., Redis): Store data as key-value pairs, optimized for fast retrieval.
* **Column Stores** (e.g., Cassandra): Organize data into columns, making them efficient for analytical queries.
* **Graph Databases** (e.g., Neo4j): Store data as nodes and edges, ideal for representing complex relationships.

Effects of NoSQL:

* **Scalability**: NoSQL databases are designed to scale horizontally, making them suitable for applications with high traffic and large data sets.
* **Flexibility**: They allow for schema-less data models, enabling rapid development and changes to data structures.
* **Performance**: Optimized for specific use cases, NoSQL databases often provide faster read/write operations for certain types of applications.
* **Eventual Consistency**: Many NoSQL databases prioritize availability and partition tolerance over immediate consistency, leading to trade-offs in data accuracy.

**Recovery Mechanism in DBMS**:

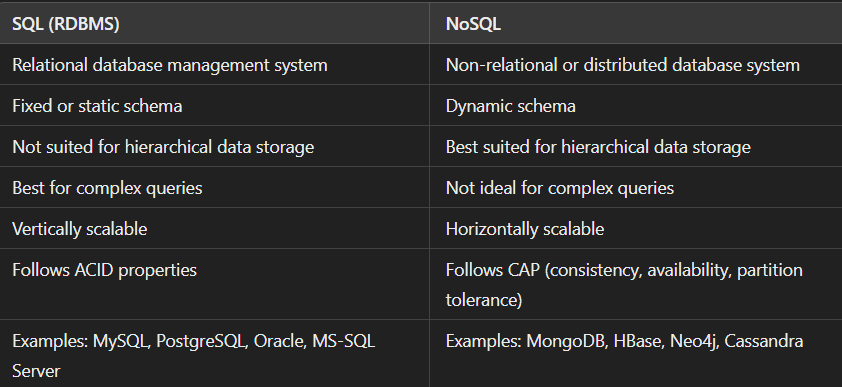
* **Shadow-Copy Scheme**: Uses copies for one active transaction; updates on commit.
* **Log-Based Recovery**: Logs transactions with deferred (post-completion) and immediate (during execution) methods.
* **Failure Handling**: Uses old values to undo failed transactions and new values for completed ones.

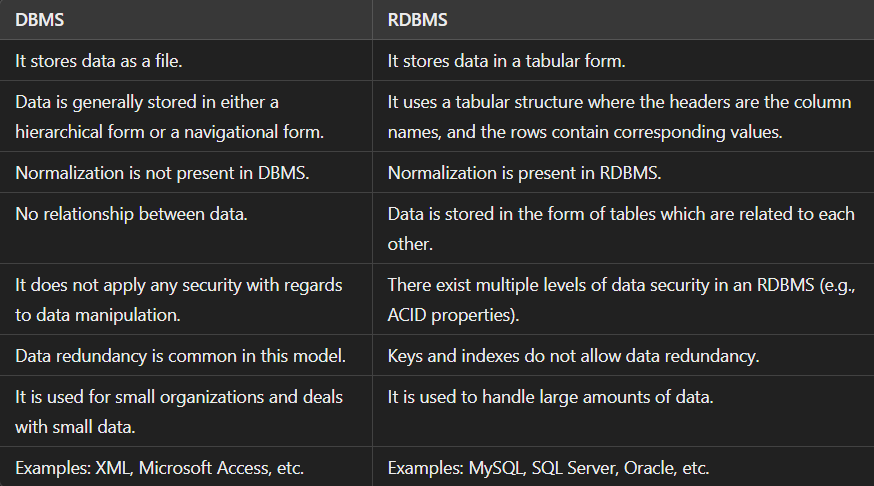
**Types of NoSQL Data Models**:

1. **Key-Value Stores**: Example: Redis
2. **Column-Oriented Stores**: Example: Cassandra
3. **Document-Based Stores**: Example: MongoDB
4. **Graph-Based Stores**: Example: Neo4j

**NoSQL Databases Disadvantages**

1. **Data Redundancy:** Can lead to larger storage due to duplication.
2. **Costly Updates/Deletes:** More resource-intensive operations.
3. **Limited Use Cases:** Not all types meet every application need.
4. **Lack of ACID Support:** Generally does not guarantee ACID properties.
5. **Inconsistent Data Entry:** Cannot enforce consistency constraints.





**Database Types**

1. **Relational Databases (RDBMS):** Use SQL; data in tables linked by foreign keys. MySQL, Oracle.
2. **Object-Oriented Databases:** Store data as objects encapsulating data and behavior. ObjectDB, GemStone.
3. **NoSQL Databases:** Flexible schema for large data; includes document, key-value, wide-column, and graph types. MongoDB, HBase.
4. **Hierarchical Databases:** Tree-like structure with single parent-child relationships. IBM IMS.
5. **Network Databases:** Allows multiple parent-child relationships in a graph structure. IDS, IDMS.

**Database clustering** combines multiple servers or instances to connect to a single database, overcoming the limitations of a single server in managing data and requests.

**Replication**: The same dataset is stored across different servers to ensure availability.

**Clustering** distributes requests across multiple servers, enabling individual processing. If one node fails, another can take over, reducing the risk of total system failure..

Advantages: Data Redundancy, Load Balancing, High Availability

**Partitioning** is the technique used to divide stored database objects into separate servers. Due to this, there is an increase in performance, controllability of the data. We can manage huge chunks of data optimally. When we horizontally scale our machines/servers, we know that it gives us a challenging time dealing with relational databases as it’s quite tough to maintain the relations. But if we apply partitioning to the database that is already scaled out i.e. equipped with multiple servers, we can partition our database among those servers and handle the big data easily.

**Sharding**

1. Technique to implement Horizontal Partitioning.

2. The fundamental idea of Sharding is the idea that instead of having all the data sit on one DB instance, we split it up and introduce a Routing layer so that we can forward the request to the right instances that actually contain the data.

**CAP Theorem :** can only guarantee two of the following three properties: Consistency, Availability, Partition Tolerance

* **CA Databases**: MySQL.
* **CP Databases**: MongoDB.
* **AP Databases**: Cassandra.

Three Schema Architecture/ **three levels of data abstraction in a DBMS**

Physical level - how data is physically stored, including storage structures and access methods (e.g., indexing, hashing), Data compression & encryption etc.

Logical level - Describes what data are stored in DB, and what relationships exist among those data.

View level –Provides a customized view of the data for end-users, hiding the complexity of the database and showing only relevant data through views.

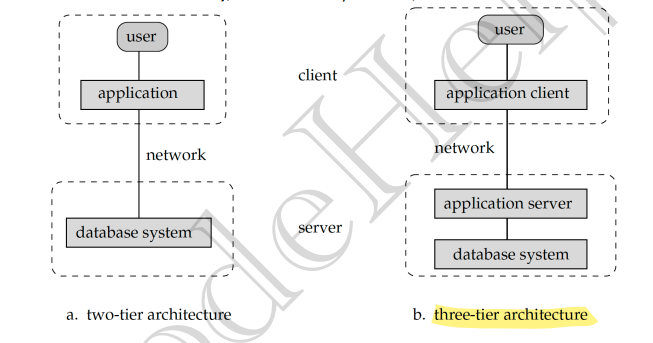
What is 2-Tier architecture?

The 2-Tier architecture is a client-server model where client applications directly communicate with the database on the server.

API standards like ODBC & JDBC are used to interact between client and server.

What is the 3-Tier architecture?

The 3-Tier architecture adds an intermediary layer, enhancing security and accessibility. The client interacts with the application server, which communicates with the database.

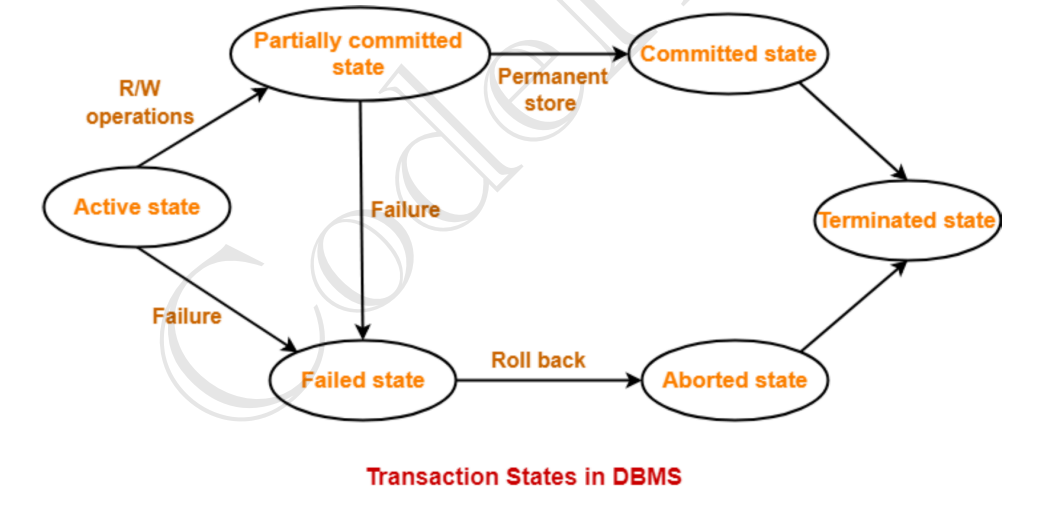


Integrity Constraints

* Domain Constraints: Restrict the Data types of every attribute.
* Entity Constraints: PK != NULL.
* Referential Constraints: If FK in referencing table refers to PK of referenced table then every value of the FK in referencing table must be NULL or available in referenced table
* Key Constraints: NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY

The **Entity-Relationship (ER) model** is a high-level conceptual data model used to design databases. It represents entities (objects or concepts), their attributes, and the relationships between these entities.

**Transaction States:**



**Schedule:** The order of operations for multiple transactions.

**Serial Schedules:** Transactions execute one after another, with no overlap. They are always consistent, recoverable, cascadeless, and strict.

**Non-Serial Schedules:** Multiple transactions execute concurrently, with operations interleaved. They may not always be consistent, recoverable, cascadeless, or strict.

**Types of Serializability:**

**Conflict Serializability:** - If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called a conflict serializable schedule. Blind write, write-read, write-write.

**View Serializability:** When a schedule is view equivalent to its serial schedule, it is said to be View Serializable, If there's no cycle in the dependency graph, it means the schedule can be transformed into a serial one, thus making it view-serializable.

**Indexing in Databases**