**Database Management 24/06**

**Database**: Collection of organized data. Stored electronically.

Features-> Data is Accessible, manageable, ability to update.  
**ANSI**: American national standard institute

**SQL**: Structured query language  
Cross platform compatibility: Diff system working software

**Rdbms**: Specific structure

Network Database: IDMS Hierarchical complex

**Different Types of Databases**

Databases can be classified in various ways, but some of the most common types include:

1. **Relational Databases (SQL Databases):**
   * **Structure:** Data is organized into tables (also called "relations") with rows and columns. Each row represents a record, and each column represents an attribute.
   * **Relationships:** Tables are linked together by common fields (keys), allowing for the establishment of predefined relationships between different data elements.
   * **Language:** They primarily use Structured Query Language (SQL) for defining, manipulating, and querying data.9
   * **Examples:** MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, IBM Db2.
   * **Use Cases:** Ideal for applications requiring strong consistency, complex queries, and transactional integrity (e.g., financial systems, e-commerce, CRM).
2. **NoSQL Databases (Non-Relational Databases):**
   * **Structure:** Designed to handle large volumes of unstructured, semi-structured, or polymorphic data. They do not rely on a fixed schema.
   * **Flexibility:** Offer high flexibility in data models, making them suitable for evolving data requirements.
   * **Scalability:** Generally designed for horizontal scalability, meaning they can distribute data across many servers.
   * **Types of NoSQL Databases:**
     + **Document Databases:** Store data in flexible, semi-structured formats like JSON, XML, or BSON documents.15 (e.g., MongoDB, Couchbase)16
     + **Key-Value Stores:** Store data as simple key-value pairs, providing fast access based on a unique key.17 (e.g., Redis, Amazon DynamoDB)18
     + **Wide-Column Stores:** Store data in column families rather than rows, suitable for analytics on specific columns.19 (e.g., Apache Cassandra, HBase)20
     + **Graph Databases:** Represent data as nodes (entities) and edges (relationships), ideal for analyzing connections and networks.21 (e.g., Neo4j, Amazon Neptune)22
   * **Use Cases:** Suitable for big data, real-time web applications, social media, IoT data, and content management systems where data models are dynamic.23
3. **Cloud Databases:**
   * **Deployment:** Stored and managed on a cloud computing platform (public, private, or hybrid cloud).24
   * **Access:** Accessed over the internet.
   * **Benefits:** Offer scalability, flexibility, managed services (automating tasks like provisioning, patching, backup), and pay-as-you-go models.25
   * **Examples:** Amazon RDS, Google Cloud SQL, Azure SQL Database.
4. **Other Notable Database Types:**
   * **Hierarchical Databases:** Organize data in a tree-like structure with parent-child relationships (one-to-many).26 (e.g., IBM IMS)
   * **Network Databases:** Similar to hierarchical but allow more complex many-to-many relationships. (e.g., IDMS)
   * **Object-Oriented Databases:** Store data in the form of objects, similar to object-oriented programming.27 (e.g., ObjectStore)
   * **In-Memory Databases:** Store data in a computer's main memory (RAM) for extremely fast data access.
   * **Time Series Databases:** Optimized for storing and analyzing time-stamped data, like sensor readings or financial market data.28
   * **Vector Databases:** Specialized for storing and searching vector embeddings, used heavily in AI applications for similarity searches.
   * **Data Warehouses:** Specifically designed for fast query and analysis of large datasets for business intelligence and reporting.29

**ACID Properties:**

**1. Atomicity**

* **Definition**: A transaction is treated as a single unit, which either **completes entirely** or **does not happen at all**.
* **Example**: In a banking system, transferring money from Account A to Account B involves two steps:
  + Deducting money from Account A
  + Adding money to Account B  
    If one step fails, the entire transaction is rolled back.

**2. Consistency**

* **Definition**: A transaction must bring the database from one **valid state to another**, maintaining all **rules and constraints**.
* **Example**: If a rule says account balances cannot be negative, a transaction that violates this rule will not be allowed to complete.

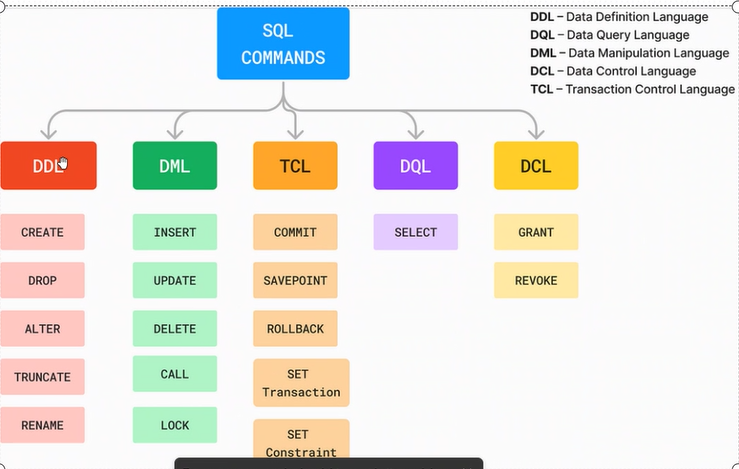
**3. Isolation**

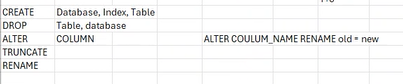
* **Definition**: Transactions are **executed independently** of one another. Intermediate states of a transaction are **not visible** to other transactions.
* **Example**: If two users are booking the last seat on a flight at the same time, isolation ensures only one booking succeeds, avoiding double-booking.

**4. Durability**

* **Definition**: Once a transaction is committed, its changes are **permanent**, even in the case of a system crash.
* **Example**: After a successful payment, the transaction record remains in the database even if the server crashes immediately afterward.

**Diff SQL Commands Classification:**





**DDL**: Statements which we use to define the schema of the table.

**Common DDL Commands:**

| **Command** | **Description** |
| --- | --- |
| CREATE | Creates a new table, database, index, or view. |
| ALTER | Modifies an existing database object (e.g., add a column). |
| DROP | Deletes an existing database object. |
| TRUNCATE | Removes all records from a table, but not the table itself. |

**DML**

**🧩 Core DML Commands**

**1. SELECT – Retrieve Data**

Used to query data from one or more tables.

SELECT column1, column2 FROM table\_name WHERE condition;

SELECT Name, Age FROM Students WHERE Age > 18;

**2. INSERT – Add New Data**

Used to insert new rows into a table.

INSERT INTO table\_name (column1, column2) VALUES (value1, value2);

INSERT INTO Students (ID, Name, Age) VALUES (1, 'Amit', 20);

**3. UPDATE – Modify Existing Data**

Used to change data in existing rows.

UPDATE table\_name SET column1 = value1 WHERE condition;

UPDATE Students SET Age = 21 WHERE Name = 'Amit';

**4. DELETE – Remove Data**

Used to delete rows from a table.

DELETE FROM table\_name WHERE condition;

DELETE FROM Students WHERE Age < 18;

| **Command** | **Action** | **Common Use Case** |
| --- | --- | --- |
| SELECT | Read data | View customer orders |
| INSERT | Add data | Add a new employee record |
| UPDATE | Modify data | Change a product price |
| DELETE | Remove data | Delete inactive user accounts |

| **Concept** | **Purpose** | **Example** | **Use Case** |
| --- | --- | --- | --- |
| CALL | Execute stored procedure | CALL UpdateSalary(1001) | Reuse business logic |
| LOCK | Control access to data | LOCK TABLE Orders IN SHARE MODE | Prevent conflicts in multi-user environments |

**DCL (Data Control Language)**

**1. GRANT**

* **Purpose**: Gives a user **permission** to perform specific actions on database objects.
* **Common Permissions**: SELECT, INSERT, UPDATE, DELETE, EXECUTE

**Example**:

GRANT SELECT, INSERT ON Students TO user123;

This allows user123 to read and insert data into the Students table.

**2. REVOKE**

* **Purpose**: **Removes permissions** previously granted to a user.

**Example**:

REVOKE INSERT ON Students FROM user123;

This removes the ability of user123 to insert data into the Students table.

**TCL (Transaction Control Language)**

| **Command** | **Action** | **Use Case** |
| --- | --- | --- |
| COMMIT | Save changes permanently | Finalize a successful transaction |
| ROLLBACK | Undo changes | Cancel a failed transaction |
| SAVEPOINT | Mark a point for partial undo | Roll back part of a transaction |
| SET TRANSACTION | Set transaction behavior | Control isolation level |



**DQL: Data Query Language  
Only SELECT statement is there.**