**1. INTRODUCTION TO RDBMS**

**1.1 What is a Database?**

A **database** is an organized collection of data that can be easily accessed, managed, and updated.  
Databases store data in a structured way so that it can be efficiently retrieved and manipulated.

**1.2 What is DBMS?**

**DBMS (Database Management System)** is software that allows users to define, create, maintain, and control access to a database.  
Examples: Microsoft Access, MySQL, Oracle, SQL Server.

**1.3 RDBMS (Relational Database Management System)**

**RDBMS** is an advanced version of DBMS that stores data in the form of **tables (relations)** consisting of **rows and columns**.  
Each table represents an **entity**, and relationships between tables are established using **keys**.

**Examples:** Microsoft SQL Server, MySQL, Oracle, PostgreSQL.

**1.4 Key Features of RDBMS**

1. Data stored in **tables** (rows and columns).
2. Each table has a **unique key** to identify records.
3. **Relationships** can be defined between tables (one-to-one, one-to-many, many-to-many).
4. Supports **SQL (Structured Query Language)** for data operations.
5. Ensures **data integrity**, **security**, and **concurrency control**.

**2. DATABASE STRUCTURE**

**2.1 Table**

A table is a database object that stores data in rows and columns.

Example Table: **Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **StudentID** | **Name** | **Age** | **Marks** |
| 1 | Udaya | 22 | 85 |
| 2 | Arun | 23 | 90 |

**2.2 Column**

Represents a field or attribute (like Name, Age, Marks).

**2.3 Row (Record)**

Represents a single data entry (e.g., one student's data).

**3. KEYS IN RDBMS**

Keys ensure **uniqueness** and define **relationships** between tables.

**3.1 Primary Key**

* Uniquely identifies each record in a table.
* Cannot contain NULL values.
* A table can have only one primary key.

**Example:**

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

Name VARCHAR(50),

Age INT

);

**3.2 Foreign Key**

* Establishes a relationship between two tables.
* References the **primary key** of another table.

**Example:**

CREATE TABLE Marks (

MarkID INT PRIMARY KEY,

StudentID INT FOREIGN KEY REFERENCES Student(StudentID),

Marks INT

);

**3.3 Candidate Key**

* All keys that can uniquely identify a record.
* One of them becomes the **primary key**.

**3.4 Composite Key**

* Combines two or more columns to uniquely identify a record.

**3.5 Unique Key**

* Ensures all values in a column are distinct, but can contain one NULL.

**3.6 Alternate Key**

* A candidate key that is not chosen as the primary key.

**4. NORMALIZATION**

**Normalization** is a process of organizing data to eliminate redundancy and improve data integrity.  
It involves dividing a large table into smaller, related tables.

**4.1 Objectives of Normalization**

1. Avoid data duplication.
2. Ensure data consistency.
3. Simplify database maintenance.
4. Establish relationships between data.

**4.2 Normal Forms**

|  |  |  |
| --- | --- | --- |
| **Normal Form** | **Rule** | **Example** |
| **1NF (First Normal Form)** | Each column holds atomic (single) values; no repeating groups. | Splitting phone numbers into separate rows. |
| **2NF (Second Normal Form)** | Should be in 1NF and all non-key attributes depend on the entire primary key. | Remove partial dependencies. |
| **3NF (Third Normal Form)** | Should be in 2NF and no transitive dependencies exist (non-key columns depend only on the key). | Split student and department data into separate tables. |

**4.3 Example of Normalization**

**Unnormalized Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **StudentID** | **StudentName** | **Dept** | **DeptHead** |
| 1 | Udaya | CS | Dr. Rao |
| 2 | Arun | CS | Dr. Rao |
| 3 | Ramesh | IT | Dr. Mehta |

**After Normalization:**

**Table 1: Student**

|  |  |  |
| --- | --- | --- |
| **StudentID** | **StudentName** | **Dept** |
| 1 | Udaya | CS |
| 2 | Arun | CS |
| 3 | Ramesh | IT |

**Table 2: Department**

|  |  |
| --- | --- |
| **Dept** | **DeptHead** |
| CS | Dr. Rao |
| IT | Dr. Mehta |

**5. SQL (STRUCTURED QUERY LANGUAGE)**

SQL is a standard language used to **store, manipulate, and retrieve data** in RDBMS.

**5.1 SQL Categories**

|  |  |  |
| --- | --- | --- |
| **Category** | **Description** | **Example** |
| **DDL (Data Definition Language)** | Defines structure of tables | CREATE, ALTER, DROP |
| **DML (Data Manipulation Language)** | Manipulates data | SELECT, INSERT, UPDATE, DELETE |
| **DCL (Data Control Language)** | Controls user access | GRANT, REVOKE |
| **TCL (Transaction Control Language)** | Manages transactions | COMMIT, ROLLBACK, SAVEPOINT |

**6. COMMON SQL COMMANDS**

**6.1 CREATE TABLE**

Creates a new table.

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

EmpName VARCHAR(50),

Salary DECIMAL(10,2),

Department VARCHAR(30)

);

**6.2 INSERT**

Inserts new records into the table.

INSERT INTO Employee (EmpID, EmpName, Salary, Department)

VALUES (1, 'Udaya', 50000, 'IT');

**6.3 SELECT**

Retrieves data from one or more tables.

SELECT EmpName, Salary FROM Employee;

Add filters and sorting:

SELECT \* FROM Employee WHERE Department = 'IT';

SELECT \* FROM Employee ORDER BY Salary DESC;

**6.4 UPDATE**

Modifies existing records.

UPDATE Employee

SET Salary = 55000

WHERE EmpID = 1;

**6.5 DELETE**

Removes records.

DELETE FROM Employee WHERE EmpID = 1;

**6.6 ALTER TABLE**

Modifies table structure.

ALTER TABLE Employee ADD Age INT;

**7. SQL CLAUSES AND OPERATORS**

**7.1 WHERE Clause**

Used to filter records based on conditions.

SELECT \* FROM Employee WHERE Salary > 40000;

**7.2 ORDER BY Clause**

Sorts the results in ascending (default) or descending order.

SELECT \* FROM Employee ORDER BY EmpName ASC;

**7.3 GROUP BY Clause**

Groups rows sharing a property so that aggregate functions can be applied.

SELECT Department, AVG(Salary) AS AvgSalary

FROM Employee

GROUP BY Department;

**7.4 HAVING Clause**

Used to filter grouped data.

SELECT Department, COUNT(\*)

FROM Employee

GROUP BY Department

HAVING COUNT(\*) > 2;

**8. JOINS IN SQL**

**Joins** combine rows from two or more tables based on related columns.

**8.1 Types of Joins**

|  |  |
| --- | --- |
| **Type** | **Description** |
| **INNER JOIN** | Returns only matching records from both tables. |
| **LEFT JOIN** | Returns all records from the left table and matched ones from the right. |
| **RIGHT JOIN** | Returns all records from the right table and matched ones from the left. |
| **FULL JOIN** | Returns all records when there is a match in either table. |
| **CROSS JOIN** | Produces a Cartesian product of both tables. |

**8.2 Example of INNER JOIN**

SELECT s.StudentName, d.DeptHead

FROM Student s

INNER JOIN Department d

ON s.Dept = d.Dept;

**9. STORED PROCEDURES**

**9.1 Definition**

A **Stored Procedure** is a precompiled group of SQL statements stored in the database.  
It improves performance and helps maintain code consistency.

**9.2 Syntax**

CREATE PROCEDURE GetEmployeeDetails

AS

BEGIN

SELECT \* FROM Employee;

END;

To execute:

EXEC GetEmployeeDetails;

**9.3 Benefits**

1. Increases performance (precompiled).
2. Reduces network traffic.
3. Enhances security (users execute procedure without direct table access).
4. Simplifies maintenance.

**10. USING SQL SERVER MANAGEMENT STUDIO (SSMS)**

**10.1 What is SSMS?**

SSMS (SQL Server Management Studio) is a graphical interface provided by Microsoft to manage SQL Server databases.

**10.2 Features**

* Create, modify, and manage databases.
* Execute SQL queries easily.
* Manage users, roles, and permissions.
* Create stored procedures and views.

**10.3 Basic Workflow**

1. Open SSMS and connect to localhost (your SQL Server instance).
2. Create a new database:
3. CREATE DATABASE DhruvTraining;
4. Create tables inside the database.
5. Insert and query data using SQL commands.
6. Create and execute stored procedures.

**SUMMARY**

|  |  |
| --- | --- |
| **Concept** | **Description** |
| **RDBMS** | Stores data in tables with relationships. |
| **Keys** | Ensure uniqueness and establish relationships. |
| **Normalization** | Removes redundancy and maintains integrity. |
| **SQL** | Language to manage and query databases. |
| **Joins** | Combine data from multiple tables. |
| **Stored Procedures** | Reusable SQL blocks stored in the database. |
| **SSMS** | Tool for managing SQL Server databases graphically. |