Create by Dhruv Prajapati~

CHAPTER 1 :

Introduction to Database System and SQL commands

Topics:

1. **Differentiate the terms: Data, Information, Records, Fields, Metadata, Data warehouse, Data dictionary.**
    a. Concepts and Definitions: Database and database systems and database environment.
    b. Data, Information, Data Item or Fields, Records, Files, Metadata, Data dictionary and it's components.
    c. Schemas, Sub-schemas, and Instances.

2. **DBMS Data types, Creating Tables (DDL), Managing Tables (DML) with SQL.**
    a. Data types.
    b. Database Language commands: Data Definition Language (DDL): CREATE, ALTER, TRUNCATE, DROP.
    c. Database Language: Data Manipulation Language (DML): INSERT, SELECT, UPDATE, DELETE.

3. **Describe & practice Transaction Control Data Control Language.**
    a. Transactional Control: Commit, Save point, Rollback.
    b. DCL Commands: Grant and Revoke.

DDL	DML	TCL	DCL
CREATE	INSERT	COMMIT	GRANT
ALTER	SELECT	ROLLBACK	REVOKE
TRUNCATE	UPDATE	SAVEPOINT	
DROP	DELETE		
RENAME			

1. Differentiate the terms: Data, Information, Records, Fields, Metadata, Data warehouse, Data dictionary.

(A.1) Concepts and Definitions: Database and database systems and database environment.

=> **1. Database** : A **database** is an organized collection of data. Imagine it like a digital library where all information is stored in a structured way so it can be easily accessed, managed, and updated.

Example: A student database might store information like names, grades, and courses.

2. Database Systems : A **Database System** is the software that manages and interacts with a database. It allows users to perform tasks like adding new data, updating existing data, and retrieving data efficiently.

Example: MySQL, Oracle, and PostgreSQL are popular database systems.

3. Database Environment : The **database environment** refers to everything that surrounds and supports a database, ensuring it functions properly. It includes the hardware, software, data, procedures, and people involved in managing the database.

(B.1)

=> **1. Data** : Data refers to raw, unprocessed facts and figures. It doesn’t have any meaningful interpretation by itself.

Example: "John," "23," "A+" (just names, numbers, or characters without context).

2. **Information** : Information is processed, organized, or structured data that is meaningful and useful for decision-making.

Example: "John is 23 years old and has an A+ blood type."

3. Data Item or Fields : A **data item** or **field** is the smallest unit of data that represents a single piece of information. Fields are the columns in a table.

Example: In a student database, fields might be **Name**, **Age**, **Grade**

4. Records : A record is a collection of related data fields that typically represent a single entity or item. Records are the rows in a table.

Example: A student record might contain the fields **Name** (John), **Age** (23), and **Grade** (A+). Together, they form one complete record.

5. Files : A file is a collection of related records that are stored together. Files typically store data for a specific purpose.

Example: A **Student File** might have multiple student records, with each record representing one student.

6. Metadata : Metadata is data about data. It provides information about the structure, format, or characteristics of the actual data. Think of it like a label that describes what the data is or how it's organized.

7. Data Dictionary : A **data dictionary** is a collection of metadata, providing detailed information about each data item, field, or table in a database. It helps define the structure of the database and provides a reference for developers and users.

Components of a Data Dictionary:

- **Field Name**: The name of the data field (e.g., "StudentName").
- **Data Type**: The type of data (e.g., text, number).
- **Field Size**: The length or size of the field (e.g., 50 characters for a name).
- **Description**: What the field is used for (e.g., "Stores the student's name").

Summary for Easy Memorization:

- **Data** = Raw facts (e.g., "John", "23").
- **Information** = Meaningful data (e.g., "John is 23 years old").
- **Fields** = Smallest unit of data (e.g., Name, Age).
- **Records** = Collection of fields (e.g., One student’s info).
- **Files** = Collection of records (e.g., All students' info).
- **Metadata** = Data about data (e.g., file size, data type).
- **Data Dictionary** = A reference that explains data (e.g., what each field is and its type).

(C.1) Understanding Schemas, Sub-schemas, and Instances.

=> **1. Schemas**

=> A **schema** is the overall structure or blueprint of a database. It defines how data is organized and how the relationships between the data are maintained.

2. Sub-schemas

=> A **sub-schema** is a part or subset of the schema that is tailored for specific users or applications. It shows only the relevant part of the database needed by a specific user or application, hiding unnecessary details.

3. Instances

=> The contents of the database at any point of time (or current state), it is referred to as **INSATANCE** of database.

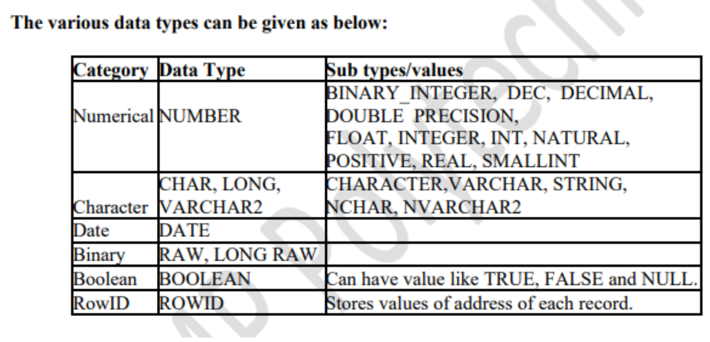
Easy Summary:

- **Schema** = Blueprint of the entire database structure (shows how everything is organized).
- **Sub-schema** = A smaller part of the schema, customized for specific users (only shows what they need).
- **Instance** = The actual data in the database at a specific time (the records that change over time).

(A.2) Data types in SQL Database.

=> In SQL, data types define the type of data that can be stored in a column. Each column in a table is assigned a specific data type, and this ensures that only valid data is stored in it.

Category	Data Type	Description	Example
Numeric	INT	Stores whole numbers (positive or negative)	100, -20
	FLOAT	Stores floating-point numbers (with decimals)	3.14, -2.718
	DECIMAL(p, s)	Stores exact numbers with p total digits and s decimals	DECIMAL(5, 2) → 123.45
	SMALLINT	Stores smaller whole numbers	50, -100
	BIGINT	Stores large whole numbers	9000000000, -1000000
	TINYINT	Stores very small whole numbers	0, 1, 255
	DOUBLE	Stores double-precision floating-point numbers	1.123456789
String	CHAR(n)	Stores fixed-length strings (padded with spaces if shorter)	CHAR(5) → 'John '
	VARCHAR(n)	Stores variable-length strings (up to n characters)	VARCHAR(50) → 'John'
	TEXT	Stores large amounts of text	"This is a long text..."
	BLOB	Stores binary data (images, videos, etc.)	ProfilePicture (image)
Date	DATE	Stores dates in YYYY-MM-DD format	'2024-09-06'
Boolean	BOOLEAN	Stores TRUE or FALSE values	TRUE, FALSE



The various data types can be given as below:

Category	Data Type	Sub types/values
Numerical	NUMBER	BINARY_INTEGER, DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, INT, NATURAL, POSITIVE, REAL, SMALLINT
Character	CHAR, LONG, VARCHAR2	CHARACTER,VARCHAR, STRING, NCHAR, NVARCHAR2
Date	DATE	
Binary	RAW, LONG RAW	
Boolean	BOOLEAN	Can have value like TRUE, FALSE and NULL.
RowID	ROWID	Stores values of address of each record.

(B.2) Database Language commands: Data Definition Language (DDL): CREATE, ALTER, TRUNCATE, DROP.

=> **Data Definition Language.**

DDL commands are used to define and manage the structure of a database.

1. **Create**

=> The CREATE command is used to create new database objects like **tables**, **databases**, etc.

=> **Syntax**

```
Create table table_name (column1 datatype(size) , ... );
```

=> **Example**

```
CREATE TABLE Students (
        StudentID Number(12),
        Name VARCHAR(50),
        Age Number(2)
);
```

2. **ALTER**

=> The ALTER command is used to modify an existing database object, like adding, deleting, or modifying columns in a table.

=> **Syntax**

```
ALTER TABLE   table_name
ADD column_name datatype;           -- To add a new column


ALTER TABLE    table_name
DROP COLUMN column_name;                 -- To remove a column


ALTER TABLE    table_name
MODIFY column_name datatype;             -- To modify a column
```

=> **Example 1 (Add Column in table)**

```
ALTER TABLE Students  ADD Email VARCHAR(100);
```

=> **Example 2 ()**

```
ALTER TABLE Students    DROP COLUMN Age;
```

3. TRUNCATE

=> The TRUNCATE command is used to **delete all data** from a table, but **keep the table structure**. It's faster than DELETE because it doesn't log individual row deletions.

=> **Syntax**

```
TRUNCATE TABLE table_name;
```

=> **Example (This removes all records from the Students table, but the table itself remains.)**

```
TRUNCATE TABLE Students;
```

4. DROP

=> The DROP command is used to **delete** an entire database object, like a table or database. Once dropped, the object cannot be recovered unless a backup exists.

=> **Syntax**

```
DROP TABLE table_name;                    -- To drop a table
DROP DATABASE database_name;              -- To drop a database
```

=> **Example 1 (Drop a table):**

```
DROP TABLE Students;
```

(C.2) Database Language: Data Manipulation Language (DML): INSERT, SELECT, UPDATE, DELETE.

=> Data Manipulation Language (DML) commands are used to manage and manipulate the data within the database

1. INSERT

The INSERT command is used to add new rows (records) to a table.

- **Syntax**:

  INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

- **Example**:

  INSERT INTO Employees (EmployeeID, FullName, Salary, HireDate) VALUES (1, 'John Doe', 50000.00, '2024-09-06');

2. SELECT

The SELECT command retrieves data from one or more tables. You can specify which columns to retrieve and filter the data.

- **Syntax**:

  SELECT column1, column2, column3, ... FROM table_name WHERE condition;

- **Example**:

  SELECT FullName, Salary FROM Employees WHERE EmployeeID = 1;

- **Example (All Columns)**:

  SELECT * FROM Employees;

3. UPDATE

The UPDATE command modifies existing records in a table based on specified conditions.

- **Syntax**:

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

- **Example**:

  UPDATE Employees SET Salary = 55000.00 WHERE EmployeeID = 1;

4. DELETE

The DELETE command removes existing records from a table based on specified conditions.

- **Syntax**:

  ```
  DELETE FROM table_name WHERE condition;
  ```

- **Example**:

  ```
  DELETE FROM Employees WHERE EmployeeID = 1;
  ```

(A.3) Transactional Control: Commit, Save point, Rollback.

=> Transactional control commands help manage transactions in SQL databases, ensuring data integrity and consistency

1. COMMIT

- **Definition**: The COMMIT command saves all changes made during the current transaction to the database. Once committed, these changes are permanent and cannot be undone.
- **Usage**: Use COMMIT to finalize a transaction after all operations are completed successfully.
- **Syntax:**

  ```
  COMMIT;
  ```

- **Example**:

  ```
  UPDATE Employees SET Salary = 60000 WHERE EmployeeID = 1;

  COMMIT;
  ```

2. SAVEPOINT

- **Definition**: The SAVEPOINT command creates a point within a transaction that you can roll back to if needed. It allows partial rollback within a transaction.
- **Usage**: Use SAVEPOINT to create restore points in a long transaction, which helps in managing complex transactions.
- **Syntax:**

  ```
  SAVEPOINT savepoint_name;
  ```

- **Example**:

  UPDATE Employees SET Salary = 60000 WHERE EmployeeID = 1;

  SAVEPOINT AfterSalaryUpdate;

  UPDATE Employees SET Salary = 70000 WHERE EmployeeID = 2;

  ROLLBACK TO AfterSalaryUpdate;

  COMMIT;

3. ROLLBACK

- **Definition**: The ROLLBACK command undoes all changes made during the current transaction or to a specific savepoint. It reverts the database to its state before the transaction began or to a specified savepoint.
- **Usage**: Use ROLLBACK to undo changes in case of errors or issues during a transaction.
- **Syntax:**

  ROLLBACK; or

  ROLLBACK TO savepoint_name;

- **Example**:

  UPDATE Employees SET Salary = 70000 WHERE EmployeeID = 2;

  ROLLBACK;

(B.3) DCL Commands: Grant and Revoke

=> Data Control Language (DCL) commands are used to control access to data in a database.

1. **GRANT**

- **Definition**: The GRANT command is used to give specific privileges (permissions) to users or roles. This can include the ability to perform certain actions on tables, views, or other database objects.
- **Usage**: Use GRANT to provide access rights to a user or a role, such as the ability to read or modify data.

- **Syntax**:

  ```
  GRANT privilege_name ON object_name TO user;
  ```

- **Example**:

  ```
  GRANT SELECT, INSERT ON Employees TO user1;
  ```

2. **REVOKE**

- **Definition**: The REVOKE command is used to remove previously granted privileges from users or roles. It effectively denies access to specified database objects.
- **Usage**: Use REVOKE to remove access rights that were previously granted, either completely or partially.
- **Syntax**:

  ```
  REVOKE privilege_type ON object_name FROM user_or_role;
  ```

- **Example**:

  ```
  REVOKE INSERT ON Employees FROM user1;
  ```