Topic 03:

Creating Tables, Inserting Data, and Constraints in SQL

In SQL, tables are the fundamental objects where data is stored in a database. You define tables using the **CREATE TABLE** statement, and you insert data using the **INSERT INTO** statement. Additionally, you can apply constraints like **NOT NULL** and **DEFAULT** to ensure data integrity.

Let's walk through each of these concepts with examples:

---

1. Create Tables

A **table** is created using the CREATE TABLE statement, followed by the table name and column definitions. Columns specify the name, data type, and any constraints for the data.

Example: Create a Table

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,              -- Unique identifier for each employee
   FirstName VARCHAR(50) NOT NULL,          -- First name of the employee, cannot be null
   LastName VARCHAR(50) NOT NULL,           -- Last name of the employee, cannot be null
   Age INT CHECK (Age >= 18),               -- Age must be 18 or older
   Department VARCHAR(50) DEFAULT 'General', -- Default department is 'General' if not provided
   HireDate DATE DEFAULT CURRENT_DATE       -- Hire date is set to the current date by default
);
```

Explanation:

- EmployeeID: This is an integer and the **primary key**, meaning it must be unique for each record.
- FirstName, LastName: Both are VARCHAR fields and cannot be **NULL**.
- Age: This field has a **CHECK** constraint, ensuring that the value is 18 or greater.
- Department: The **DEFAULT** value for the department is set to 'General' if no value is provided.
- HireDate: The **DEFAULT** value for the HireDate is set to the current date (CURRENT_DATE).

---

2. Insert Data into the Tables

Data can be inserted into a table using the INSERT INTO statement. There are two common methods for inserting data: **Single Insert** and **Bulk Insert**.

---

3. Single Insert

The **single insert** allows you to add one row of data at a time.

Example: Single Insert

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, HireDate)
VALUES (1, 'John', 'Doe', 30, 'Engineering', '2024-01-15');
```

Explanation:

- The INSERT INTO statement specifies the table name (Employees).
- The column names are provided within parentheses, followed by the values to be inserted into those columns.
- This command inserts a single row with the specified data.

---

4. Bulk Insert

The **bulk insert** allows you to insert multiple rows of data in one INSERT INTO statement, which is more efficient than inserting one row at a time.

Example: Bulk Insert

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, HireDate)
VALUES
(2, 'Jane', 'Smith', 25, 'Marketing', '2024-02-10'),
(3, 'Sam', 'Johnson', 35, 'Finance', '2024-03-01'),
(4, 'Emily', 'Davis', 28, 'HR', '2024-04-20');
```

Explanation:

- Multiple sets of values are provided, each enclosed in parentheses and separated by commas.
- Each set of values corresponds to a row to be inserted into the table.

---

5. Not Null Constraint

The **NOT NULL** constraint ensures that a column cannot have a NULL value. This is useful when you want to enforce that a column always contains a value.

Example: Adding Not Null Constraint

When creating a table, you can use the NOT NULL constraint for one or more columns. This was shown in the **Create Table** example above for FirstName and LastName.

However, you can also add a NOT NULL constraint to an existing column using the ALTER TABLE statement.

Example: Alter Table to Add NOT NULL Constraint

```
ALTER TABLE Employees
MODIFY COLUMN Department VARCHAR(50) NOT NULL;
```

Explanation:

- This command modifies the Department column in the Employees table to **NOT NULL**, ensuring that every employee must have a department value.

---

6. Default Constraint

The **DEFAULT** constraint provides a default value for a column when no value is specified during the **INSERT** operation. If a value is not provided, the default is used.

Example: Using the Default Constraint

In the **Create Table** example above, the Department and HireDate columns had default values:

- Department defaults to 'General'.
- HireDate defaults to the current date (CURRENT_DATE).

Example: Inserting Data with Default Values

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age)
VALUES (5, 'David', 'Wilson', 40);
```

Explanation:

- Since no value is provided for the Department and HireDate columns, the default values will be used:
  - Department will default to 'General'.
  - HireDate will default to the current date (e.g., '2024-12-22').

You can verify this by querying the table:

```
SELECT * FROM Employees;
```

---

7. Example: Full Workflow of Creating, Inserting, and Applying Constraints

Let's combine all the concepts in one complete example. We will create a table with constraints, insert single and bulk records, and see how **NOT NULL** and **DEFAULT** constraints work.

```
-- Step 1: Create the Table with Constraints
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,              -- Primary key for uniqueness
    FirstName VARCHAR(50) NOT NULL,          -- Not null constraint
    LastName VARCHAR(50) NOT NULL,           -- Not null constraint
    Age INT CHECK (Age >= 18),               -- Check constraint for age >= 18
    Department VARCHAR(50) DEFAULT 'General', -- Default constraint for department
    HireDate DATE DEFAULT CURRENT_DATE        -- Default constraint for hire date
);

-- Step 2: Insert a Single Row of Data
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age)
VALUES (1, 'John', 'Doe', 30);

-- Step 3: Insert Multiple Rows of Data (Bulk Insert)
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, HireDate)
VALUES
(2, 'Jane', 'Smith', 25, 'Marketing', '2024-02-10'),
(3, 'Sam', 'Johnson', 35, 'Finance', '2024-03-01'),
(4, 'Emily', 'Davis', 28, 'HR', '2024-04-20');

-- Step 4: Query the Table to See the Data
SELECT * FROM Employees;
```

Explanation:

1. The Employees table is created with constraints such as NOT NULL, DEFAULT, and CHECK.
2. We inserted a **single row** for John Doe without specifying the Department and HireDate (they will use the default values).
3. We performed a **bulk insert** for three other employees, each specifying their Department and HireDate.
4. A SELECT query retrieves the data, showing that default values have been applied where needed.

Conclusion

1. **Creating Tables**: Use CREATE TABLE to define the structure of a table, specifying columns, data types, and constraints like NOT NULL, DEFAULT, and CHECK.
2. **Inserting Data**:
   - Use **single inserts** for one row at a time.
   - Use **bulk inserts** for multiple rows to improve performance.
3. **Constraints**:
   - NOT NULL ensures a column cannot be empty.
   - DEFAULT assigns a default value when no value is provided during an insert.

Mastering these SQL operations is essential for effectively managing relational databases and ensuring data integrity through constraints.