**SQL: Data Definition Commands (DDC)**

**Introduction to Data Definition Commands (DDC)**

* Data Definition Commands (DDC) are used to define the structure of a database. They focus on the creation, modification, and removal of database objects such as tables, views, and constraints.
* DDCs are key to setting up and maintaining the schema of a database.

**1. Creating Table Structure**

The CREATE TABLE command is used to create a new table, which defines the structure and types of data the table will hold.

**Basic Syntax:**

sql

CREATE TABLE table\_name (

column\_name datatype [constraints],

column\_name datatype [constraints],

...

);

* column\_name: Name of the column.
* datatype: Defines the type of data the column will store (e.g., VARCHAR, INT, DATE).
* constraints: Optional rules like PRIMARY KEY, NOT NULL, etc.

**Example:**

sql

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50),

BirthDate DATE,

Salary DECIMAL(10, 2),

DepartmentID INT

);

* In this example, the Employees table is created with a PRIMARY KEY on EmployeeID and a NOT NULL constraint on FirstName.

**2. Data Types in SQL**

Data types define what kind of data can be stored in each column. Commonly used data types include:

**Numeric Types:**

* INT: Integer numbers.
* DECIMAL(p, s): Fixed-point numbers, p is precision (total number of digits), s is scale (number of digits after the decimal point).
* FLOAT: Floating-point numbers.

**Character Types:**

* VARCHAR(n): Variable-length character strings, where n is the maximum length.
* CHAR(n): Fixed-length character strings.

**Date and Time Types:**

* DATE: Stores date (year, month, day).
* TIME: Stores time (hours, minutes, seconds).
* DATETIME: Stores both date and time.

**Other Types:**

* BOOLEAN: Stores TRUE or FALSE.
* BLOB: Stores binary large objects, e.g., images or files.

**3. Implementing SQL Constraints**

SQL constraints are rules applied to the columns or tables to ensure data integrity.

**a. PRIMARY KEY**

* Uniquely identifies each record in a table.
* Automatically creates a unique index for the column(s).
* Cannot accept NULL values.

sql

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

**b. FOREIGN KEY**

* Establishes a link between two tables by enforcing a relationship between columns.
* Ensures referential integrity (i.e., values in the foreign key column must exist in the referenced table).

sql

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**c. UNIQUE**

* Ensures that all values in a column are distinct.

sql

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Email VARCHAR(100) UNIQUE

);

**d. NOT NULL**

* Ensures a column does not accept NULL values.

sql

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL

);

**e. CHECK**

* Ensures values in a column satisfy a specific condition.

sql

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

Price DECIMAL(10, 2) CHECK (Price > 0)

);

**4. Changing Column Data Types and Characteristics**

You can modify the characteristics of an existing column using the ALTER TABLE command.

**a. Changing Column Data Type**

Use MODIFY COLUMN or ALTER COLUMN (depending on the RDBMS) to change a column's data type.

sql

ALTER TABLE Employees

MODIFY COLUMN Salary DECIMAL(12, 2);

**b. Changing Column Constraints**

You can add or remove constraints from an existing column.

sql

ALTER TABLE Employees

MODIFY COLUMN Name VARCHAR(100) NOT NULL;

**5. Adding and Dropping Columns**

You can add or remove columns from an existing table using ALTER TABLE.

**a. Adding a Column**

sql

ALTER TABLE table\_name ADD column\_name datatype [constraints];

* **Example**:

sql

ALTER TABLE Employees ADD Email VARCHAR(100);

**b. Dropping a Column**

sql

ALTER TABLE table\_name DROP COLUMN column\_name;

* **Example**:

sql

ALTER TABLE Employees DROP COLUMN BirthDate;

**6. Adding, Removing, Enabling, and Disabling Constraints**

SQL provides flexibility to manage constraints during the lifecycle of a table.

**a. Adding Constraints**

Use ALTER TABLE to add constraints to an existing table.

sql

ALTER TABLE table\_name

ADD CONSTRAINT constraint\_name constraint\_type (column\_name);

* **Example**:

sql

ALTER TABLE Employees

ADD CONSTRAINT FK\_Department FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID);

**b. Dropping Constraints**

You can drop a constraint with ALTER TABLE using the DROP CONSTRAINT clause.

sql

ALTER TABLE table\_name DROP CONSTRAINT constraint\_name;

* **Example**:

sql

ALTER TABLE Employees

DROP CONSTRAINT FK\_Department;

**c. Enabling/Disabling Constraints**

Some databases allow you to enable or disable constraints for maintenance purposes.

* **Enable**:

sql

ALTER TABLE table\_name ENABLE CONSTRAINT constraint\_name;

* **Disable**:

sql

ALTER TABLE table\_name DISABLE CONSTRAINT constraint\_name;

**7. Creating Virtual Tables (Views)**

Views are virtual tables based on the result set of a query. They don't store data physically but act as a window to query data from underlying tables.

**a. Simple Views**

A simple view is based on a single table or a subset of columns in a table.

sql

CREATE VIEW SimpleEmployeeView AS

SELECT FirstName, LastName, Salary

FROM Employees

WHERE Salary > 50000;

**b. Complex Views**

A complex view involves multiple tables, often using JOIN, GROUP BY, or subqueries.

sql

CREATE VIEW ComplexEmployeeView AS

SELECT e.FirstName, e.LastName, d.DepartmentName

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE d.DepartmentName = 'Sales';

**Benefits of Views:**

* Simplifies complex queries.
* Provides an abstraction layer to hide the complexity of underlying table structures.
* Can be used to present data securely by limiting the columns exposed.

**8. Summary**

* **CREATE**: Used to define new objects like tables, views, and indexes.
* **Data Types**: Defines the type of data (e.g., INT, VARCHAR, DATE) that can be stored in a column.
* **Constraints**: Ensure data integrity and consistency through rules like PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK.
* **ALTER**: Used to modify the structure of an existing table, including adding or removing columns and constraints.
* **Views**: Virtual tables that simplify complex queries and provide an abstraction layer.

These Data Definition Commands are foundational for creating and maintaining the structure and integrity of a database system.