Foreign Key Constraints

Objective: Ensure referential integrity between tables by linking columns in one table to the primary key or unique key of another table.

What is a Foreign Key?

A **Foreign Key (FK)** is a column or a combination of columns that establishes a link between the data in two tables. It ensures that the value in the foreign key column matches a value in the referenced table's primary key or a unique column.

Syntax:

```
sql
Copy
FOREIGN KEY (foreign_key_column) REFERENCES referenced_table
(referenced_column)
```

Example 1: Creating a Foreign Key Constraint

Consider two tables: `Orders` and `Customers`. The `Orders` table will contain a `CustomerID` column that references the `CustomerID` in the `Customers` table.

```
sql
Copy
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    Email VARCHAR(100)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

In this example:

- The `Orders` table has a `CustomerID` column, which is a **foreign key** that references the `CustomerID` in the `Customers` table.
- This ensures that every value in the `CustomerID` of `Orders` must match an existing value in the `CustomerID` of the `Customers` table.

Example 2: Foreign Key with ON DELETE CASCADE

You can also specify actions to be taken when a referenced row in the parent table is deleted or updated. For example, when a customer is deleted from the `Customers` table, all orders associated with that customer in the `Orders` table will also be deleted automatically.

```
sql
Copy
```

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerID)
        REFERENCES Customers(CustomerID)
        ON DELETE CASCADE
);
```

In this case, if a row in the `Customers` table is deleted, all corresponding rows in the `Orders` table will be deleted due to the **ON DELETE CASCADE** action.

Check Constraints

Objective: Enforce domain integrity by limiting the range of values that can be inserted into a column.

What is a Check Constraint?

A **Check Constraint** is used to ensure that all values in a column satisfy a specified condition. It can be applied to a single column or multiple columns.

Syntax:

```
sql
Copy
CHECK (condition)
```

Example 1: Creating a Check Constraint

In the following example, we enforce a **Check Constraint** to ensure that the `Age` column in the `Employees` table always contains values greater than or equal to 18.

```
sql
Copy
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Age INT,
    Salary DECIMAL(10, 2),
    CHECK (Age >= 18)
);
```

Here, the `CHECK (Age >= 18)` constraint ensures that only employees aged 18 or older can be inserted into the `Employees` table.

Example 2: Check Constraint on Multiple Columns

You can also define a **Check Constraint** that applies to multiple columns. For example, ensuring that an employee's `Salary` must be greater than `Age * 1000`:

```
sql
Copy
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Age INT,
    Salary DECIMAL(10, 2),
    CHECK (Salary > (Age * 1000))
);
```

This ensures that for every employee, their salary is greater than their age multiplied by 1000.

DDL (Data Definition Language) Commands

Objective: Use DDL commands to define and manage database structure.

What is DDL?

DDL commands are used to define, modify, and remove database objects like tables, indexes, and schemas. Key DDL commands include:

1. **CREATE**: Used to create a new database object like a table or view.
2. **ALTER**: Used to modify an existing database object.
3. **DROP**: Used to delete an existing database object.
4. **TRUNCATE**: Used to remove all records from a table without removing the table structure.

1. CREATE Table Command

The `CREATE` command is used to create new tables or other database objects.

Syntax:

```
sql
Copy
CREATE TABLE table_name (
    column1 datatype [constraint],
    column2 datatype [constraint],
    ...
);
```

Example 1: Creating a Table

```
sql
Copy
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Age INT,
    EnrollmentDate DATE
);
```

Here, we create a `Students` table with a primary key `StudentID` and other columns for student data.

2. ALTER Table Command

The `ALTER` command allows modification of an existing table. This includes adding, deleting, or modifying columns.

Syntax:

```
sql
Copy
ALTER TABLE table_name
ADD column_name datatype;
```

Example 1: Adding a New Column

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

This adds a new column `Email` to the `Students` table.

Example 2: Modifying an Existing Column

```
sql
Copy
ALTER TABLE Students
MODIFY COLUMN Age INT NOT NULL;
```

This modifies the `Age` column to make it **NOT NULL**.

Example 3: Dropping a Column

```
sql
Copy
ALTER TABLE Students
DROP COLUMN EnrollmentDate;
```

This drops the `EnrollmentDate` column from the `Students` table.

3. DROP Table Command

The `DROP` command removes an existing table and all of its data from the database.

Syntax:

```
sql
Copy
DROP TABLE table_name;
```

Example:

```
sql
Copy
DROP TABLE Students;
```

This deletes the `Students` table from the database.

4. TRUNCATE Command

The `TRUNCATE` command is used to remove all rows from a table but keeps the table structure intact. It's more efficient than `DELETE` when deleting all rows.

Syntax:

```
sql
Copy
TRUNCATE TABLE table_name;
```

Example:

```
sql
Copy
TRUNCATE TABLE Students;
```

This deletes all rows in the `Students` table but leaves the table definition intact.

Summary of Key Points

- **Foreign Key Constraints** ensure referential integrity between tables, linking columns to primary or unique keys in other tables.
- **Check Constraints** enforce domain integrity, ensuring that values in a column satisfy certain conditions.
- **DDL Commands** are used to define and manage database objects. These include `CREATE`, `ALTER`, `DROP`, and `TRUNCATE`.

These commands and constraints play a critical role in maintaining data integrity and structuring the database. They allow developers to create flexible, reliable, and consistent database systems.