Constraints in BDSM

Aim: To understand and apply various SQL constraints to ensure data integrity and consistency within a relational database.

Course Outcomes:

- **CO2:** Design database for an organization using the relational model.
- **CO3:** Apply data manipulation language to query and update the database. (partially covered by demonstrating data insertion with constraints)

Theory:

SQL constraints are rules implemented on database tables to ensure data accuracy and reliability. They define limitations on the data types and values allowed within a table, preventing invalid data entry and maintaining the integrity of relationships between tables. Constraints can be applied at the column level or the table level, depending on the specific restriction.

Types of Constraints:

- **NOT NULL:** Enforces that a column cannot contain a NULL value.
- **UNIQUE:** Ensures that all values within a column are unique.
- **PRIMARY KEY:** A combination of NOT NULL and UNIQUE, uniquely identifying each row in a table.
- **FOREIGN KEY:** Maintains referential integrity between tables by preventing actions that would break the link between a child table and a parent table.
- **CHECK:** Limits the range of values allowed in a column based on a specified condition.
- **DEFAULT:** Sets a default value for a column if no value is provided during insertion.

Syntax:

- **NOT NULL:** `column_name datatype NOT NULL`
- **UNIQUE:** `column_name datatype UNIQUE`
- **PRIMARY KEY:** `CONSTRAINT constraint_name PRIMARY KEY (column_name)`
- **FOREIGN KEY:** `CONSTRAINT constraint_name FOREIGN KEY (column_name) REFERENCES parent_table(column_name)`
- **CHECK:** `CONSTRAINT constraint_name CHECK (condition)`
- **DEFAULT:** `column_name datatype DEFAULT default_value`

Example:

SQL

```
CREATE TABLE Employees (
    EmployeeID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL UNIQUE,
    FirstName VARCHAR(255),
    Age INT CHECK (Age >= 18),
    City VARCHAR(255) DEFAULT 'Unknown',
    DepartmentID INT,
    PRIMARY KEY (EmployeeID),
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

Learning Outcomes:

Upon completion of this experiment, students will be able to:

- Define and explain the purpose of various SQL constraints.
- Implement SQL constraints during table creation or modification.
- Differentiate between column-level and table-level constraints.
- Understand the importance of constraints in maintaining data integrity.
- Apply constraints to enforce data validation rules.
- Troubleshoot common errors related to constraint violations.

code

```
-- Creating a new table called 'Employees'
CREATE TABLE Employees (
    EmployeeID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255),
    Age INT,
    City VARCHAR(255)
);

-- Adding a PRIMARY KEY constraint to the EmployeeID column
ALTER TABLE Employees
ADD PRIMARY KEY (EmployeeID);

-- Adding a UNIQUE constraint to the LastName column
ALTER TABLE Employees
ADD UNIQUE (LastName);

-- Adding a CHECK constraint to the Age column
ALTER TABLE Employees
```

```
ADD CHECK (Age >= 18);

-- Adding a DEFAULT constraint to the City column
ALTER TABLE Employees
ALTER COLUMN City SET DEFAULT 'Unknown';

-- Creating a new table called 'Departments'
CREATE TABLE Departments (
    DepartmentID INT NOT NULL,
    DepartmentName VARCHAR(255) NOT NULL,
    PRIMARY KEY (DepartmentID)
);

-- Adding a FOREIGN KEY constraint to the Employees table
ALTER TABLE Employees
ADD DepartmentID INT;

ALTER TABLE Employees
ADD FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID);

-- Inserting data into the Employees table
INSERT INTO Employees (EmployeeID, LastName, FirstName, Age, City,
DepartmentID) VALUES
(1, 'Smith', 'John', 30, 'London', 1),
(2, 'Jones', 'Sarah', 25, 'Paris', 2),
(3, 'Williams', 'David', 40, 'New York', 1);

-- Inserting data into the Departments table
INSERT INTO Departments (DepartmentID, DepartmentName) VALUES
(1, 'Sales'),
(2, 'Marketing');
```