Practical 4: Creating Employee Table with Constraints

Aim: Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique.

Code:

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) UNIQUE

);

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

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

DeptID INT REFERENCES Department(DeptID)

);

-- Insert Valid Data

INSERT INTO Department (DeptID, DeptName) VALUES (1, 'HR');

INSERT INTO Department (DeptID, DeptName) VALUES (2, 'IT');

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Alice', 'alice@example.com', 50000.00, 1);

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (102, 'Bob', 'bob@example.com', 60000.00, 2);

-- Insert Invalid Data to Test Constraints

Duplicate Primary Key

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (101, 'Charlie', 'charlie@example.com', 55000.00, 1);

-- Duplicate Unique Email

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (103, 'David', 'alice@example.com', 45000.00, 2);

-- Salary Check Constraint Violation

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES (105, 'Frank', 'frank@example.com', -40000.00, 1);

Practical 5: Testing Employee Constraints

Aim: To test constraints like PRIMARY KEY, UNIQUE, and CHECK by inserting invalid data into the Employee table.

Code:

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Age INT CHECK (Age >= 18),

IsActive BOOLEAN DEFAULT TRUE

);

-- Insert Valid Data

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (1, 'John', 'Doe', 'john.doe@example.com', '1234567890', 25, TRUE);

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES (2, 'Jane', 'Smith', 'jane.smith@example.com', '0987654321', 30);

-- Insert Invalid Data to Test Constraints

-- Invalid data for NOT NULL constraint (FirstName is NULL)

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES (3, NULL, 'Taylor', 'taylor@example.com', '5551234567', 20);

-- Invalid data for CHECK constraint (Age less than 18)

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES (4, 'Alice', 'Johnson', 'alice.johnson@example.com', '6669876543', 16);

-- Invalid data for UNIQUE constraint (Duplicate Email)

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES (5, 'Bob', 'Brown', 'john.doe@example.com', '7771234567', 28);

Practical 6:

Aim: Use DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes.

Code:

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Department VARCHAR(50),

Salary DECIMAL(10, 2)

);

(DML Command)

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)

VALUES (1, 'John', 'Doe', 28, 'HR', 50000.00);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)

VALUES (2, 'Jane', 'Smith', 35, 'IT', 65000.00);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)

VALUES (3, 'Michael', 'Johnson', 40, 'Finance', 75000.00);

Updates (DML Commands)

-- 1. Update a single column (e.g., update salary for EmployeeID 2)

UPDATE Employees

SET Salary = 70000.00

WHERE EmployeeID = 2;

-- 2. Update multiple columns for a specific row (e.g., update name and salary for EmployeeID 2)

UPDATE Employees

SET FirstName = 'Janet', LastName = 'Williams', Salary = 75000.00

WHERE EmployeeID = 2;

-- 3. Update entire tuple (all columns for EmployeeID 3)

UPDATE Employees

SET FirstName = 'Michael', LastName = 'Brown', Age = 45, Department = 'Management', Salary = 80000.00

WHERE EmployeeID = 3;

-- 4. Update with a condition (e.g., increase salary by 10% for all employees in HR)

UPDATE Employees

SET Salary = Salary \* 1.10

WHERE Department = 'HR';

-- 5. Update with a subquery (e.g., increase salary for Employee with highest salary)

UPDATE Employees

SET Salary = Salary + 5000

WHERE Salary = (SELECT MAX(Salary) FROM Employees);

-- 6. Update using a CASE statement (e.g., increase salary based on department)

UPDATE Employees

SET Salary = CASE

WHEN Department = 'HR' THEN Salary \* 1.05

WHEN Department = 'IT' THEN Salary \* 1.08

WHEN Department = 'Finance' THEN Salary \* 1.10

ELSE Salary

END;

-- Delete Data from the Table (DML Command)

DELETE FROM Employees

WHERE EmployeeID = 1;

-- Select and Verify Data (SELECT Query)

-- To retrieve all data from the table

SELECT \* FROM Employees;

-- To verify the update (checking updated values for EmployeeID 2)

SELECT \* FROM Employees

WHERE EmployeeID = 2;

-- To verify the deletion (checking if EmployeeID 1 exists)

SELECT \* FROM Employees

WHERE EmployeeID = 1;