08.09.2025

Task 5 : Writing join queries, equivalent, and / or recursive queries.

Aim : To implement and execute JOIN queries, equivalent queries, and recursive queries using a university database scenario.

Procedure :

The SQL joins clause is used to combine records from two or more tables in a database. A Join is a means for combining fields from two tables by using values common to each. The join is actually performed by the 'where' clause which combines specified rows of tables.

create the database and tables (~~students,~~ departments, courses, enrollments).

Insert sample data

write SQL queries using different types of JOINS.

write equivalent queries (different approaches to get the same result).

Implement a recursive query (using WITH RECURSIVE)

Display results and verify correctness.

Syntax:-

```
SELECT column 1, column 2, column 3... FROM
table-name1, table-name2 WHERE table-name1.
column name = table-name2. columnname;
```

Different Types of SQL JOINS

(INNER) JOIN :

```
INNER JOIN table 2 ON table 1. column - name =
table 2. column - name;
```

LEFT (OUTER) JOIN :

```
LEFT JOIN table 2 ON table 1. column - name =
table 2. column - name;
```

RIGHT (OUTER) JOIN :

```
RIGHT JOIN table 2 ON table 1 . colum - name =
table 2. column - name;
```

FULL (OUTER) JOIN :

```
FULL OUTER JOIN table 2 ON table 1. column - name =
table 2. column - name;
```

1. JOIN QUERIES (All Types)

CREATE TABLES

```
CREATE TABLE Departments (
Dept ID INT PRIMARY KEY,
Dept Name VARCHAR (50)
);
```

```
CREATE TABLE Students (
Student ID INT PRIMARY KEY,
Student Name VARCHAR (50),
Dept ID INT,
FOREIGN KEY (Dept ID) REFERENCES Departments
(Dept ID)
);

CREATE TABLE Courses (
Course ID VARCHAR (10) PRIMARY KEY,
Course Name VARCHAR (50),
Dept ID INT,
FOREIGN KEY (Dept ID) REFERENCES Departments
(Dept ID)
);

CREATE TABLE Enrollments (
EnrollID INT PRIMARY KEY,
Student ID INT,
Course ID VARCHAR (10),
FOREIGN KEY (student ID) REFERENCES students
(student ID),
FOREIGN KEY (course ID) REFERENCES Courses
(course ID)
);
```

```
CREATE TABLE Prerequisites (
Course ID VARCHAR (10),
PrereqID VARCHAR (10)
);
```

2. INSERT SAMPLE DATA

```
INESERT INTO Departments VALUES
(101, 'Computer Science'),
(102, 'Electrical Science'),
(103, 'Mechanical Engg');

INSERT INTO Students VALUES
(1, 'Alice', 101),
(2, 'Bob', 102),
(3, 'Charlie', 101),
(4, 'David', 103),
(5, 'Emma', 104); -- Invalid DeptID for OUTER
JOIN example

INSERT INTO Courses VALUES
('C1', 'Database systems', 101),
('C2', 'Operating systems', 101),
('C3', 'Circuits', 102),
('C4', 'Thermodynamics', 103);
```

```
Insert Into Enrollments VALUES
(1,1,'C1'),
(2,1,'C2'),
(3,2,`C3'),
(4,3,'C1'),
(5,4,`C4').

INSERT INTO Prerequisities VALUES
('C2',`C1') , -- OS requires DB
('C3',`C2') ; -- Circuits requires OS.
```

3. JOIN QUIERES (ALL TYPES)

a) INNER JOIN

```
SELECT S. Student Name , d. deptName
FROM Students.S
INNER JOIN Departments d ON S.DeptID = d.DeptID;
```

b) LEFT JOIN

```
SELECT S. StudentName , d. deptName
FROM student S.
LEFT JOIN Departments d ON S. DeptID = d.DeptID;
```

c) RIGHT JOIN

```
SELECT S.StudentName , d.DeptName
FROM Student S
RIGHT JOIN Departments d ON S. DeptID = d. DeptID;
```

d) FULL OUTER JOIN (Postgres SQL / oracle only ; not in MYSQL)

```
SELECT s.StudentName , d.DeptName
FROM Student s
FULL OUTER JOIN Departments d ON s.deptID = deptID;
```

e) CROSS JOIN

```
SELECT s.StudentName , c.CourseName
FROM Students s
CROSS JOIN Courses c;
```

f) SELF JOIN

```
SELECT S1.StudentName AS Student1 , S2.StudentName AS
Student2 , S1.DeptID
FROM Students S1
JOIN Students S2 ON S1.DeptID = S2.DeptID
WHERE S1.StudentID < S2.StudentID
```

4. EQUIVALENT QUERIES

Using JOIN

```
SELECT s.StudentName , d.deptName
FROM Students s
JOIN Departments d ON s.DeptID = d.DeptID.
```

--- Using subquery

```
SELECT StudentName.
    (SELECT DeptName FROM Departments of WHERE d.DeptID=
S DeptID) AS DeptName
FROM Student S;
```

5. RECURSIVE QUERY (Course Hierarchy)

```
WITH RECURSIVE Courselitoarchy AS (
SELECT Course ID , PrereqID
FROM Prerequistes
UNION
SELECT p.courseID , c.prereqID
FROM prerequistes p
JOIN courseliterarchy c ON p.PrereqID = c.courseID
)
SELECT * FROM Courseliterarchy;
```

8/9 Result : The Implementation of SQL commands using Joins and recursive queries are executed successfully.