Handling Join queries. Equivalent, and/or Recursive queries

Aim:-

To implement and execute Join queries, Equivalent queries and Recursive queries. ~~using a views~~

Procedure:

+ create the database and tables(Patient, Department, Doctor, Appointment)

+ Insert sample data)

+ write SQL queries using different types of Joins

+ write equivalent queries

+ Implement a recursive query

+ Display results and verify correctness.

Step 1

Syntax

SELECT column1, column2, column2, column3 From table-name1, table_name2. where table_name1.column name = table_name2.column name;

Different types of sql Joins

Inner Join:

tables SELECT column-names(s)

Feom table Inner join table2 on table1.

Left (outer) Join:

SELECT column-name(s) From table1
Left Join table2 ON table1.column-name
= table2.colum-name.

RIGHT (outer) JOIN:

SELECT column_name(s) from table1
RIGHT JOIN table2 ON table1.column_name
= table2.column_name;

FULL (outer) JOIN:

FULL outer Join table2 on table1.colum
n_name = table2.column_name;

1. JOIN Queries (All types)

Create table Departments(
DEPT_ID INT PrimaryKEY,
Dept Name VARCHAR(50));

Create Table patients(
PatID INT PRIMARY KEY,
DeptName VARCHAR(50),
DeptID INT,
FOREIGN KEY(DeptID) References
departments (DeptID);

```
Create Table Bills
    Bill ID INT PRIMARY KEY,
    Pat ID INT,
    BillAmount DECIMAL(10,2);
    FOREIGN KEY(Pat ID) REFFREUES
    Patients(PatID));
```

```
CREATE TABLE Referrals(
    ReferralID INT PRIMARY KEY,
    ReferringPatID INT,
    ReferredPatID INT,
    FOREIGN KEY(ReferringPatID) Reffers
    Patients(PatID));
```

2. Departments:

DepAID	DeptName

Patients

PatID	PatName	DeptID

Table Bills

BillID	DatID	BillAmount

Referrals

ReferralID	ReferringPatID	ReferredPatID

2 Insert sample data

Insert into Departments values (101,'cardiology'),
(102,'oncology'), (103,'Pediatrics');

Insert into patient values
(1, 'Alice', 101);
(2, BOB, 102),
(3, charlie, 101),
(4, David, 103),
(5, emma, 101));

Insert Into Bills values(
(1, 1, 5000.00),
(2, 1, 300.00),
(3, 2, 1500.00),
(4, 3, 6000.00),);

Insert into References values(
(1, 1, 3);
(2, 3, 4);

3. JOIN Queries (All types)

a) Inner Join

query:

select P.Patname, d.DeptName
From Patients P Inner JOIN Departments
d on P.DepartID = d.DeptID;

PatName	DeptName
Alice	cardiology
BOB	oncology
charlie	cardiology
David	Pediatrics

b) left JOIN

SELECT P.PatName, d.DeptName
from Patients P left JOIN Departments
d ON D.DeptId = d.dept.ID;

Output

PatName	Dept Name
Alice	cardiology
Bob	oncology
charlie	cardiology
David	Pediatrics
Emma	NULL

c) Right JOIN

Query:

```
SELECT P.PatName, d.deptName
From Patient p Right join Department
d ON P.DeptID = d.deptID
```

Output:

PatName	DeptName
Alice	cardiology
BoB	oncology
Charlie	cardiology
David	Pediatrics

d) Full outer Join

```
Select P.PatName, d.DeptName
From Patientsp FULL outer JOIN
Departments d ON P.DeptID = d.
DeptID;
```

output

PatName	DeptName
Alice	cardiology
Bob	oncology
Charlie	cardiology
David	Pediatrics
emma	NULL

c) self join

SELECT P1.PatName AS Patient1, P2.PatName AS Patient2 P1.DeptID From PatientsD1 JOIN PatientsD2 ON D1.DeptID = D2 DEPTID Where D1.PatID < D2.PatID;

Output

Patient1	Patient 2	Dept ID
Alice	Charlie	101

4. Equivalent queries

using joins: Join vs subquery

Select D.PatName, d.DeptName from PatientsD JOIN Departments, d ON D.DeptID = d.DeptID

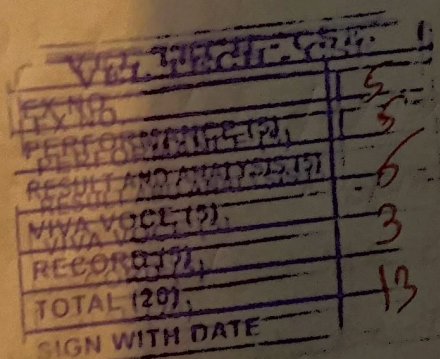
using subqueries:

Select PatName (select DeptName From Departments d where d.Dept = D.DeptID As DeptName From Patients P;

Output

Pat Name	Deptname
Alice	Cardiology
BOB	Oncology
Charlie	Cardiology
David	Pediatrics

5th 8/9/25



Result:

Thus the implementation of circular queue, queue using linked list and recursive algorithms are executed successfully.