Agenda

- Joins

Table Relations

- To avoid redundancy of the data, data should be organized into multiple tables so that tables are related to each other.
- The relations can be one of the following
    1. One to One
    2. One to Many
    3. Many to One
    4. Many to Many
- Entity relations is outcome of Normalization process.
- We need to apply Multi-Table Joins

Joins

- It is used to fetch the data from multiple tables in single query
- Types of joins are
    1. CROSS JOIN
    2. INNER JOIN
    3. LEFT OUTER JOIN
    4. RIGHT OUTER JOIN
    5. FULL OUTER JOIN
    6. SELF JOIN

```
-- use joins.sql for all the join demos
SOURCE <path to joins.sql file>

SELECT * FROM emps;
SELECT * FROM depts;
SELECT * FROM addr;
SELECT * FROM meeting;
SELECT * FROM emp_meeting;
```

1. CROSS JOIN

```
SELECT ename,dname FROM emps CROSS JOIN depts;

SELECT e.ename,d.dname FROM emps e CROSS JOIN depts d;

SELECT e.ename,d.dname FROM depts d CROSS JOIN emps e;

SELECT e.ename,d.dname FROM emps AS e CROSS JOIN depts AS d;
-- using AS in table alias is optional
```

```
SELECT emps.ename,depts.dname FROM emps CROSS JOIN depts;
-- can use table name directly if table alias are not given
```

2. INNER JOIN

- The inner JOIN is used to return rows from both tables that satisfy the join condition.
- Non-matching rows from both tables are skipped.
- If join condition contains equality check, it is referred as equi-join,otherwise it is non-equi-join.

```
-- display empname and his deptname
SELECT e.ename,d.dname FROM emps e
INNER JOIN depts d ON e.deptno=d.deptno;
-- called as equi-joins

-- display empname and the deptnames in which he is not working
SELECT e.ename,d.dname FROM emps e
INNER JOIN depts d ON e.deptno!=d.deptno;
-- called as non-equi joins
```

LEFT OUTER JOIN

- Common data from both table, plus the data from left side table
- Left outer join is used to return matching rows from both tables along with additional rows in left table.
- Corresponding to additional rows in left table, right table values are taken as NULL.

```
-- display empname and his deptname.
SELECT e.ename,d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno=d.deptno;

SELECT e.ename,d.dname FROM emps e
LEFT JOIN depts d ON e.deptno=d.deptno;
-- using outer keyword is optional
```

RIGHT OUTER JOIN

- Common data from both table, plus the data from right side table
- Right outer join is used to return matching rows from both tables along with additional rows in right table.
- Corresponding to additional rows in right table, left table values are taken as NULL.

```
-- display empname and his deptname.
SELECT e.ename,d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno=d.deptno;
```

```
SELECT e.ename,d.dname FROM emps e
RIGHT JOIN depts d ON e.deptno=d.deptno;
-- using outer keyword is optional
```

Full Outer Join

- Full join is used to return matching rows from both tables along with additional rows in both tables.
- Corresponding to additional rows in left or right table, opposite table values are taken as NULL.
- Full outer join is not supported in MySQL, but can be simulated using set operators.

```
SELECT e.ename, d.dname FROM emps e
FULL OUTER JOIN depts d ON e.deptno = d.deptno;
-- NOT SUPPORTED
```

Set Operators

- Used to combine results of two queries (if output contains same number of columns).

```
(SELECT dname AS name FROM depts)
UNION ALL
(SELECT ename FROM emps);

(SELECT sal FROM emp)
UNION ALL
(SELECT price FROM books);

(SELECT e.ename, d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno = d.deptno)
UNION ALL
(SELECT e.ename, d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno = d.deptno);

(SELECT e.ename, d.dname FROM emps e
LEFT OUTER JOIN depts d ON e.deptno = d.deptno)
UNION
(SELECT e.ename, d.dname FROM emps e
RIGHT OUTER JOIN depts d ON e.deptno = d.deptno);
-- simulation of full outer join in MySQL
```

Self Join

```
-- display ename and manager name from emps;
SELECT e.ename, m.ename AS mname FROM emps e
INNER JOIN emps m ON e.mgr = m.empno;
```

```
SELECT e.ename, m.ename AS mname FROM emps e
LEFT JOIN emps m ON e.mgr = m.empno;
```

Order of clauses in queries

```
SELECT cols FROM table1
JOIN table2
WHERE condition
GROUP BY column
HAVING condition
ORDER BY column
LIMIT n;
```