MySQL - RDBMS

Agenda

- Sub-queries
- Views

Q & A

```
SELECT deptno, SUM(sal) FROM emp
WHERE job != 'MANAGER'
GROUP BY deptno;
```

```
UPDATE dept d
INNER JOIN emp e ON e.deptno = d.deptno
SET d.dname = 'ACCOUNTING'
WHERE e.ename = 'KING';

SELECT * FROM dept;
```

```
SELECT d.deptno, d.dname, SUM(e.sal) FROM emp e
INNER JOIN dept d ON e.deptno = d.deptno
GROUP BY d.deptno, d.dname;
```

Assignment 6

```
SELECT o.onum, c.cname FROM orders o
INNER JOIN customers c ON o.cnum = c.cnum;
```

```
SELECT o.onum, c.cname, s.sname FROM orders o
INNER JOIN customers c ON o.cnum = c.cnum
INNER JOIN salespeople s ON o.snum = s.snum;
```

```
-- wrong query
SELECT o.onum, c.cname, s.sname FROM orders o
INNER JOIN customers c ON o.cnum = c.cnum
INNER JOIN salespeople s ON c.snum = s.snum;
```

```
SELECT c.cname, s.sname, s.comm FROM customers c
INNER JOIN salespeople s ON c.snum = s.snum
WHERE s.comm > 0.12;
```

```
SELECT o.onum, s.sname, o.amt * s.comm FROM orders o
INNER JOIN customers c ON o.cnum = c.cnum
INNER JOIN salespeople s ON o.snum = s.snum
WHERE c.rating > 100;
```

```
SELECT s1.snum snum1, s1.sname sname1, s2.snum s2, s2.sname sname2 FROM
salespeople s1
INNER JOIN salespeople s2 ON s1.city = s2.city
WHERE s1.snum < s2.snum;
```

RDBMS Preparations

- Interview preparations -- SQL queries -- Joins, Sub-queries
  - Assignments
  - Assesments
  - Hackerrank -- SQL -- Easy & Medium.
- CCEE preparations -- MCQ
  - Moodle MCQ activity
    - Data entry till end of module -- After that read-only
    - Search functionality -- Topicwise, Wordwise.
- Lab exams
  - Lab Assignments & Assesments

Sub-queries

Single row sub-queries

- Return single row from inner query.

```
USE classwork;

-- display emp with max sal.

SELECT * FROM emp ORDER BY sal DESC LIMIT 1;
-- will produce partial output if multiple emps have same max sal.

SELECT * FROM emp WHERE sal = MAX(sal);
-- error: Group fns cannot be used in WHERE clause

SET @maxsal = (SELECT MAX(sal) FROM emp);
```

```
SELECT @maxsal;
SELECT * FROM emp WHERE sal = @maxsal;

SELECT * FROM emp
WHERE sal = (SELECT MAX(sal) FROM emp);

-- display emp with second highest sal
SELECT * FROM emp ORDER BY sal DESC LIMIT 1, 1;

SET @sal2 = (SELECT DISTINCT sal FROM emp ORDER BY sal DESC LIMIT 1,1);
SELECT @sal2;
SELECT * FROM emp WHERE sal = @sal2;

SELECT * FROM emp WHERE sal = (SELECT DISTINCT sal FROM emp ORDER BY sal DESC
LIMIT 1,1);

-- display emp with third highest sal
SELECT * FROM emp WHERE sal = (SELECT DISTINCT sal FROM emp ORDER BY sal DESC
LIMIT 2,1);

-- display emps working in dept of 'KING'
SET @dno = (SELECT deptno FROM emp WHERE ename = 'KING');
SELECT * FROM emp WHERE deptno = @dno;

SELECT * FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'KING');

SELECT * FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'KING') AND
ename != 'KING';
```

Multi-row sub-queries

- Inner query returns multiple rows.
- They can be compared using ANY, ALL or IN operator.

```
-- find all emps having sal more than all salesman.
SELECT sal FROM emp WHERE job = 'SALESMAN';

SELECT MAX(sal) FROM emp WHERE job = 'SALESMAN';

SELECT * FROM emp WHERE sal > (SELECT MAX(sal) FROM emp WHERE job = 'SALESMAN');

SELECT * FROM emp WHERE sal >
ALL(SELECT sal FROM emp WHERE job = 'SALESMAN');
-- (sal > 1600 AND sal > 1250 AND sal > 1250 AND sal > 1500)
```

```
-- find emp with sal less than sal of "any" emp in deptno=20
SELECT sal FROM emp WHERE deptno = 20;

SELECT * FROM emp WHERE sal < (SELECT MAX(sal) FROM emp WHERE deptno = 20);
```

```
SELECT * FROM emp WHERE sal < ANY(SELECT sal FROM emp WHERE deptno = 20);
-- sal < 800 OR sal < 2975 OR sal < 3000 OR sal < 1100 OR sal < 3000.
```

```
-- display the depts which have emps.
SELECT deptno FROM emp;

SELECT * FROM dept WHERE deptno = ANY(SELECT deptno FROM emp);
-- deptno = 10 OR deptno = 20 OR deptno = 30

SELECT * FROM dept WHERE deptno IN (SELECT deptno FROM emp);
-- deptno = 10 OR deptno = 20 OR deptno = 30
```

- ANY vs IN operator
    - ANY can be used in sub-queries only, while IN can be used with/without sub-queries.
    - ANY can be used for comparision (<, >, <=, >=, =, or !=), while IN can be used only for equality comparision (=).
    - Both operators are logically similar to OR operator.
- ANY vs ALL operator
    - Both can be used for comparision (<, >, <=, >=, =, or !=).
    - Both are usable only with sub-queries.
    - ANY is similar to logical OR, while ALL is similar to logical AND.

```
-- display depts which do not have any emp.

SELECT * FROM dept WHERE deptno NOT IN (SELECT deptno FROM emp);
-- deptno != 10 AND deptno != 20 AND deptno != 30
-- NOT (deptno = 10 OR deptno = 20 OR deptno = 30)

SELECT * FROM dept WHERE deptno != ALL(SELECT deptno FROM emp);
```

Corelated sub-queries

- SELECT ... FROM table WHERE col = (SELECT ...)
- By default inner query is executed for each row of the outer query.
- If no optimization settings are enabled, sub-queries are slower than joins.

```
SELECT * FROM dept WHERE deptno IN (SELECT deptno FROM emp);
-- 10, ACC --> SELECT deptno FROM emp
-- 20, RES --> SELECT deptno FROM emp
-- 30, SAL --> SELECT deptno FROM emp
-- 40, OPS --> SELECT deptno FROM emp
```

- The sub-query execution can be speed-up if inner queries return/process less number rows.

```
SELECT * FROM dept WHERE deptno IN (SELECT DISTINCT deptno FROM emp);
```

- This is typically done by using WHERE clause in inner query.
- The WHERE clause in inner query may depend on current row of the outer query. This kind of query is called as "co-related sub-query".

```
SELECT * FROM dept d WHERE d.deptno IN
(SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- 10, ACC --> SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno --> 3 rows
(10,10,10)
-- 20, RES --> SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno --> 5 rows
(20,20,20,20,20)
-- 30, SAL --> SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno --> 6 rows
(30,30,30,30,30,30)
-- 40, OPS --> SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno --> 0 rows
```

```
SELECT * FROM dept d WHERE EXISTS
(SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- EXISTS check if sub-query return row(s).
```

```
-- display all depts which do not contain any emp.
SELECT * FROM dept d WHERE NOT EXISTS
(SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
```

Sub-query in projection

```
-- display number of emps in each dept along with total number of emps.
SELECT deptno, COUNT(empno) FROM emp
GROUP BY deptno;

SELECT COUNT(empno) FROM emp;

(SELECT deptno, COUNT(empno) FROM emp
GROUP BY deptno)
UNION
(SELECT NULL, COUNT(empno) FROM emp);

-- +--------+--------------+
-- | deptno | COUNT(empno) |
-- +--------+--------------+
-- |     20 |            5 |
```

```
-- |     30 |            6 |
-- |     10 |            3 |
-- |   NULL |           14 |
-- +--------+--------------+

SELECT deptno,
COUNT(empno) AS deptcnt,
(SELECT COUNT(empno) FROM emp) AS totalcnt
FROM emp
GROUP BY deptno;
-- +--------+---------+----------+
-- | deptno | deptcnt | totalcnt |
-- +--------+---------+----------+
-- |     20 |       5 |       14 |
-- |     30 |       6 |       14 |
-- |     10 |       3 |       14 |
-- +--------+---------+----------+
```

Sub-query in FROM clause

- Inner-query can be written in FROM clause of SELECT statement. The output of the inner query is treated as a table (MUST give table alias) and outer query execute on that table.
- This is called as "Derived table" or "Inline view".

```
-- display empno, ename, sal and category of emp.
-- < 1500 -> POOR, 1500-2500 -> MID, > 2500 -> RICH
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal BETWEEN 1500 AND 2500 THEN 'MIDDLE'
ELSE 'RICH'
END AS category
FROM emp;
```

```
-- display number of emps in each category.
SELECT category, COUNT(empno) FROM
(SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal BETWEEN 1500 AND 2500 THEN 'MIDDLE'
ELSE 'RICH'
END AS category
FROM emp) AS emp_category
GROUP BY category;
```

Sub-query in DML

```
-- INSERT new emp with name 'JOHN' and sal=2000 in dept 'OPERATIONS'.
SELECT deptno FROM dept WHERE dname = 'OPERATIONS';

INSERT INTO emp(ename, sal, deptno) VALUES ('JOHN', 2000, (SELECT deptno FROM dept
WHERE dname = 'OPERATIONS'));

SELECT * FROM emp;
```

```
-- give comm 100 to all emps in OPERATIONS dept.
UPDATE emp SET comm = 100 WHERE deptno =
(SELECT deptno FROM dept WHERE dname = 'OPERATIONS');

-- UPDATE emp SET comm = 100 WHERE deptno = 40;

SELECT * FROM emp;
```

```
-- delete emps from OPERATIONS dept.
DELETE FROM emp WHERE deptno =
(SELECT deptno FROM dept WHERE dname = 'OPERATIONS');

-- DELETE FROM emp WHERE deptno = 40;
```

- In MySQL, DML cannot be performed on the table from which inner query is selecting.

```
INSERT INTO emp(empno,ename,sal) VALUES(1000, 'JACK', 6000);

-- delete emp with max sal.
DELETE FROM emp
WHERE sal = (SELECT MAX(sal) FROM emp);
-- error: not allowed in MySQL

SET @maxsal = (SELECT MAX(sal) FROM emp);
DELETE FROM emp WHERE sal = @maxsal;
```

SQL Performance

- Modern RDBMS implement lot of optimization mechanisms (internally based on data structures and algorithms) like caching (materialization), semijoins or hashjoins, etc.
- These optimization logic will differ from RDBMS to RDBMS.

  ```
  SELECT @@optimizer_switch;
  ```

- In MySQL, query Performance is measured in terms of query cost. Lower the cost, better is performance.
- The cost of query depends on data in the table(s), MySQL version, server machine config, optimizer settings, etc.

```
EXPLAIN FORMAT=JSON
SELECT * FROM emp WHERE sal < (SELECT MAX(sal) FROM emp WHERE deptno = 20);
-- 1.65

EXPLAIN FORMAT=JSON
SELECT * FROM emp WHERE sal < ANY(SELECT sal FROM emp WHERE deptno = 20);
-- 1.65
```

Views

- View is projection of the data.
- CREATE VIEW viewname AS SELECT ...;
- In MySQL views are non-materialized i.e. output of SELECT statement (of view) is not saved in server disk. Only SELECT query is saved in server disk.

```
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal BETWEEN 1500 AND 2500 THEN 'MIDDLE'
ELSE 'RICH'
END AS category
FROM emp;

CREATE VIEW v_empcategory AS
SELECT empno, ename, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal BETWEEN 1500 AND 2500 THEN 'MIDDLE'
ELSE 'RICH'
END AS category
FROM emp;

SHOW TABLES;

SHOW FULL TABLES;

SELECT * FROM v_empcategory;

SELECT category, COUNT(empno) FROM v_empcategory
GROUP BY category;

EXPLAIN FORMAT=JSON
SELECT category, COUNT(empno) FROM v_empcategory
GROUP BY category;
```

```
SHOW CREATE VIEW v_empcategory;

SELECT * FROM v_empcategory;

ALTER VIEW v_empcategory AS
SELECT empno, sal, CASE
WHEN sal < 1500 THEN 'POOR'
WHEN sal BETWEEN 1500 AND 2500 THEN 'MIDDLE'
ELSE 'RICH'
END AS category
FROM emp;

SELECT * FROM v_empcategory;

DROP VIEW v_empcategory;

SHOW FULL TABLES;
```

```
CREATE VIEW v_empsal AS
SELECT empno, ename, sal FROM emp;

SELECT * FROM v_empsal;

CREATE VIEW v_richemp AS
SELECT * FROM emp WHERE sal > 2500;

SELECT * FROM v_richemp;

CREATE VIEW v_empincome AS
SELECT empno, ename, sal, comm, sal + IFNULL(comm,0.0) income FROM emp;

SELECT * FROM v_empincome;

CREATE VIEW v_empjobsummary AS
SELECT job, SUM(sal) salsum, AVG(sal) salavg, MAX(sal) salmax, MIN(sal) salmin
FROM emp
GROUP BY job;

SELECT * FROM v_empjobsummary;

INSERT INTO emp(empno,ename,job,sal,deptno) VALUES (1000, 'JILL', 'DEV', 2000,
40);

SELECT * FROM v_empjobsummary;
```

```
SELECT * FROM v_empsal;

INSERT INTO v_empsal VALUES (1001, 'JOY', 2300);
```

```
-- DML ops are allowed on Simple Views.

SELECT * FROM emp;
```

```
SELECT * FROM v_empjobsummary;

DELETE FROM v_empjobsummary WHERE job = 'DEV';
-- DML ops are NOT allowed on Complex Views.
```