MySQL - RDBMS

Agenda

- Sub-queries
- Views
- ALTER
- Indexes
- Constraints

Q & A

```
SELECT emp.ename, dept.dname FROM emp
INNER JOIN dept ON emp.deptno = dept.deptno;

SELECT ename, dname FROM emp
INNER JOIN dept ON emp.deptno = dept.deptno;
```

```
SELECT e.ename, d.dname FROM emp e
INNER JOIN dept d ON e.deptno = d.deptno;

SELECT e.ename, d.dname FROM emp e
INNER JOIN dept d USING (deptno);
-- join on common column -- deptno.

SELECT e.ename, d.dname FROM emp e
NATURAL JOIN dept d;
-- inner join on common column -- names of common columns are found automatically.
```

Sub-queries

Single-Row Sub-query

```
-- find emp with max sal.
SELECT * FROM emp WHERE sal = MAX(sal);
-- error: group fn cannot be used in WHERE clause

SET @maxsal = (SELECT MAX(sal) FROM emp);
-- get output of max sal query into a variable @maxsal.
SELECT @maxsal;
-- print the variable @maxsal
SELECT * FROM emp WHERE sal = @maxsal;
-- use the variable in where clause to get desired result.

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

```
-- instead of a var using a query in another query -- sub-query

-- find the last hired employee in company.
SELECT * FROM emp
ORDER BY hire DESC
LIMIT 1;

SET @lasthire = (SELECT MAX(hire) FROM emp);
SELECT @lasthire;
SELECT * FROM emp WHERE hire = @lasthire;

SELECT * FROM emp WHERE hire = (SELECT MAX(hire) FROM emp);
```

```

-- find emp with highest sal.
SELECT * FROM emp ORDER BY sal DESC LIMIT 1;

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

-- find emp with 3rd highest sal.
SELECT * FROM emp ORDER BY sal DESC;

SELECT * FROM emp ORDER BY sal DESC LIMIT 2, 1;

SELECT DISTINCT sal FROM emp ORDER BY sal DESC;

SELECT DISTINCT sal FROM emp ORDER BY sal DESC LIMIT 2,1;

SET @sal3 = (SELECT DISTINCT sal FROM emp ORDER BY sal DESC LIMIT 2,1);
SELECT @sal3;
SELECT * FROM emp WHERE sal = @sal3;

SELECT * FROM emp WHERE sal = (SELECT DISTINCT sal FROM emp ORDER BY sal DESC
LIMIT 2,1);

-- find emp with 2nd highest sal.
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);

-- find all emps working in dept of JAMES.
SELECT deptno FROM emp WHERE ename = 'JAMES';

SET @james_dept=(SELECT deptno FROM emp WHERE ename = 'JAMES');
SELECT * FROM emp WHERE deptno = @james_dept;

SELECT * FROM emp
WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'JAMES');
-- WHERE clause & hence sub-query is executed once for each record.
-- For emp table it will be executed 14 times (because there are 14 rows).

-- However most of RDBMS have optimizations, which ensure that sub-query result is
cached (when appropriate)
-- This execute sub-query only once and hence much efficient,
-- This depends on RDBMS optimizations.

SELECT e.ename, e.deptno FROM emp e
INNER JOIN emp j ON e.deptno = j.deptno
WHERE j.ename = 'JAMES';

-- optimization settings of MySQL can be seen in optimizer_switch variable.
SELECT @@optimizer_switch;
```

Single-Row Sub-query

```
-- find all emps having sal more than sals of salesman.

-- using single row sub-query
SELECT MAX(sal) FROM emp WHERE job = 'SALESMAN';

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

-- using multi-row sub-query
SELECT sal FROM emp WHERE job = 'SALESMAN';

SELECT * FROM emp
WHERE sal > (SELECT sal FROM emp WHERE job = 'SALESMAN');
-- error: left side of > have one value, right side of > have 4 values.
-- multi-row sub-query results cannot be compared using relational operators

SELECT * FROM emp
WHERE sal > ALL(SELECT sal FROM emp WHERE job = 'SALESMAN');
-- ALL keyword -- compare with all values.
```

```
-- find all emps having sal less than than any of the salesman.

SELECT sal FROM emp WHERE job = 'SALESMAN';

-- using single row sub-query
SELECT MAX(sal) FROM emp WHERE job = 'SALESMAN';

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

-- using multi-row sub-query
SELECT sal FROM emp WHERE job = 'SALESMAN';
SELECT * FROM emp
WHERE sal < ANY(SELECT sal FROM emp WHERE job = 'SALESMAN');
-- ANY keyword -- compare with all values and find if sal is less than any of
them.

-- find all emps whose sal is less than sal of all salesman.
SELECT * FROM emp
WHERE sal < ALL(SELECT sal FROM emp WHERE job = 'SALESMAN');
```

```
-- find all depts which have some emp.

SELECT * FROM emp;

SELECT * FROM dept;

SELECT deptno FROM emp;

SELECT * FROM dept
WHERE deptno = (SELECT deptno FROM emp);
-- error: left side have one deptno and right side have 14 deptno

SELECT * FROM dept
WHERE deptno = ANY(SELECT deptno FROM emp);

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

Multi-Row Sub-query Operators

- IN operator / NOT IN operator
  - To check equality in multi-row sub-queries.
- ALL operator
  - comparing with all values and if condition is true for all of them, then consider it true.
  - similar to logical AND.
  - SELECT sal FROM emp WHERE job = 'SALESMAN';
    - 1600, 1250, 1250, 1500.
  - SELECT * FROM emp WHERE sal > ALL(SELECT sal FROM emp WHERE job = 'SALESMAN');
    - sal > 1600 AND sal > 1250 AND sal > 1250 AND sal > 1500.
- ANY operator
  - comparing with all values and if condition is true for any of them, then consider it true.
  - similar to logical OR.
  - SELECT sal FROM emp WHERE job = 'SALESMAN';
    - 1600, 1250, 1250, 1500.
  - SELECT * FROM emp WHERE sal > ANY(SELECT sal FROM emp WHERE job = 'SALESMAN');
    - sal > 1600 OR sal > 1250 OR sal > 1250 OR sal > 1500.

Correlated Sub-query

- Inner query is executed for each row of outer query.
- To improve performace of sub-queries, the inner query should return minimum number of rows.

```
SELECT * FROM dept
WHERE deptno IN (SELECT deptno FROM emp);
-- since outer query has 4 rows, inner query will be executed 4 times.

SELECT * FROM dept
WHERE deptno IN (SELECT DISTINCT deptno FROM emp);
-- even though sub-query returns only 3 rows, but it need to process all 14 rows
to find unique.

SELECT deptno FROM emp WHERE deptno = 10; --> 3 rows -- 10, 10, 10
SELECT deptno FROM emp WHERE deptno = 20; --> 5 rows -- 20, 20, 20, 20, 20
SELECT deptno FROM emp WHERE deptno = 30; --> 6 rows -- 30, 30, 30, 30, 30, 30

SELECT * FROM dept d
WHERE d.deptno IN (SELECT e.deptno FROM emp e);
-- still inner query returns 14 rows only.
-- dept table have 4 rows, so sub-query executed 4 times and returns 14 rows each
times.

SELECT * FROM dept d
WHERE d.deptno IN (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- dept table have 4 rows, so sub-query executed 4 times.
--      SELECT e.deptno FROM emp e WHERE e.deptno = 10; -- 3 rows
--      SELECT e.deptno FROM emp e WHERE e.deptno = 20; -- 5 rows
--      SELECT e.deptno FROM emp e WHERE e.deptno = 30; -- 6 rows
--      SELECT e.deptno FROM emp e WHERE e.deptno = 40; -- 0 rows
-- here sub-query returns less rows, hence better performance.
-- condition is based on current row of outer query.

SELECT * FROM dept d
WHERE EXISTS (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- exists only check if inner query result is non-empty.

-- find all depts in which there are no employees
SELECT * FROM dept d
WHERE d.deptno NOT IN (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);

SELECT * FROM dept d
WHERE NOT EXISTS (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- exists only check if inner query result is empty.
```

Sub-query with DML

```
-- delete emp with max sal.
DELETE FROM emp WHERE sal = MAX(sal);
-- error: group function in WHERE clause is not allowed.
```

```
DELETE FROM emp
WHERE sal = (SELECT MAX(sal) FROM emp);
-- error: not supported in MySQL.
-- MySQL doesn't allow inner query to be performed on the table, in which
UPDATE/DELETE operation is going on.

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

SELECT * FROM emp;

-- delete dept in which there is no emp.
DELETE FROM dept
WHERE deptno NOT IN (SELECT deptno FROM emp);
-- alowed in MySQL
-- Table to delete from: "dept", Table on which inner query is running: "emp".

SELECT * FROM dept;

-- insert JOHN in RESEARCH dept as ANALYST on sal 2800.
INSERT INTO emp(empno, ename, job, sal, deptno)
VALUES (1000, 'JOHN', 'ANALYST', 2800, 20);

INSERT INTO emp(empno, ename, job, sal, deptno)
VALUES (1000, 'JOHN', 'ANALYST', 2800,
    (SELECT deptno FROM dept WHERE dname='RESEARCH')
);

SELECT * FROM emp;

-- insert JOHN in RESEARCH dept as ANALYST on with sal same as highest ANALYST
sal.
-- keep its empno as MAX(empno) + 1.

INSERT INTO emp(empno, ename, job, sal, deptno)
VALUES (
    (SELECT MAX(empno) FROM emp)+1,
    'JOHN',
    'ANALYST',
    (SELECT MAX(sal) FROM emp WHERE job = 'ANALYST'),
    (SELECT deptno FROM dept WHERE dname='RESEARCH')
);
-- error: Not supported in MySQL
-- Homework: Do this using variables
```

Query Performance

```
EXPLAIN FORMAT=JSON
SELECT * FROM dept d
WHERE d.deptno IN (SELECT e.deptno FROM emp e);
```

```
-- 4.90

EXPLAIN ANALYZE
SELECT * FROM dept d
WHERE d.deptno IN (SELECT e.deptno FROM emp e);

EXPLAIN FORMAT=JSON
SELECT * FROM dept d
WHERE EXISTS (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
-- 4.90

EXPLAIN ANALYZE
SELECT * FROM dept d
WHERE EXISTS (SELECT e.deptno FROM emp e WHERE e.deptno = d.deptno);
```