MySQL - RDBMS

Agenda

- Logical vs Physical Layout
- MySQL data types
- CHAR vs VARCHAR vs TEXT
- SQL scripts
- DQL - SELECT
    - Projection
    - Computed Columns
    - DISTINCT
    - LIMIT
    - ORDER BY
    - WHERE
        - Relational Operators
        - NULL Operators
        - Logical Operators

DBMS

- DBMS -- Database Management (CRUD)
    - Traditional databases - File based databases
    - Relational databases
    - NoSQL databases

Logical vs Physical Layout

MySQL data types

```
* CREATE TABLE stud(id INT, name CHAR(20), marks DOUBLE);
    * id is by default signed.
* CREATE TABLE stud(id INT UNSIGNED, name CHAR(20), marks DECIMAL(5,2));
    * id is now unsigned.
    * DECIMAL(5,2) -- total 5 digits and 2 digits after decimal point.
```

```sql
CREATE TABLE temp(c1 CHAR(4), c2 VARCHAR(4), c3 TEXT(4));

DESCRIBE temp;

INSERT INTO temp VALUES('abcd', 'abcd', 'abcdef');

INSERT INTO temp VALUES('xy', 'xy', 'xy');

INSERT INTO temp VALUES('pqr', 'pqr', 'pqr');
```

```
INSERT INTO temp VALUES('pqrst', 'pqr', 'pqr');
-- error

INSERT INTO temp VALUES('pqr', 'pqrst', 'pqr');
-- error

SELECT * FROM temp;
```

CHAR vs VARCHAR vs TEXT

SQL scripts

```
USE classwork;

SELECT USER(), DATABASE();
-- sunbeam@localhost, classwork

SOURCE D:/classwork-db.sql

SHOW TABLES;

SELECT * FROM books;

SELECT * FROM dept;
```

DQL - SELECT

- Projection -- Display only given columns.

```
SELECT * FROM dept;
-- * all columns

SELECT deptno, dname FROM dept;

SELECT * FROM emp;

SELECT empno, ename, sal FROM emp;

SELECT sal, ename, deptno FROM emp;

CREATE TABLE newemp(id INT, name CHAR(30), sal DOUBLE);

INSERT INTO newemp SELECT * FROM emp;
-- error

INSERT INTO newemp SELECT empno, ename, sal FROM emp;

SELECT * FROM newemp;
```

```
INSERT INTO newemp(id, name, sal) SELECT empno, ename, sal FROM emp;

SELECT * FROM newemp;
```

```
-- display ename, sal, da=sal*0.5, gs=sal+sal*0.5
SELECT ename, sal, sal*0.5, sal+sal*0.5 FROM emp;
-- computed columns or pseudo columns.

SELECT ename AS name, sal, sal * 0.5 AS da, sal + sal * 0.5 AS gs FROM emp;
-- column alias

SELECT ename name, sal, sal * 0.5 da, sal + sal * 0.5 gs FROM emp;
-- writing AS is optional

SELECT ename name, sal, sal * 0.5 da, sal + sal * 0.5 gross sal FROM emp;
-- error

SELECT ename name, sal, sal * 0.5 da, sal + sal * 0.5 `gross sal` FROM emp;
```

```
-- display ename, deptno and dname (10=ACCOUNTING, 20=RESEARCH, 30=SALES).
SELECT ename, deptno FROM emp;

SELECT ename, deptno, CASE
WHEN deptno = 10 THEN 'ACCOUTING'
WHEN deptno = 20 THEN 'RESEARCH'
WHEN deptno = 30 THEN 'SALES'
ELSE 'UNKNOWN'
END AS deptname
FROM emp;
```

```
SELECT DISTINCT job FROM emp;

SELECT DISTINCT deptno FROM emp;

SELECT deptno, job FROM emp;
-- 10 --> CLERK, MANAGER, PRESIDENT
-- 20 --> CLERK, ANALYST, MANAGER
-- 30 --> CLERK, MANAGER, SALESMAN

SELECT DISTINCT deptno, job FROM emp;
-- displays unique combination of deptno & job.
```

```
DESCRIBE emp;
```

```
SELECT * FROM emp;
-- show all rows

SELECT * FROM emp LIMIT 5;
-- show first 5 rows.

SELECT * FROM emp LIMIT 10;
-- show first 10 rows.

SELECT ename, sal FROM emp LIMIT 10;
-- show ename & sal of first 10 rows.

SELECT ename, sal FROM emp LIMIT 5;
-- show ename & sal of first 5 rows.

SELECT ename, sal FROM emp LIMIT 3, 5;
-- show ename & sal of 5 rows after first 3 rows.
```

```
SELECT * FROM emp;

SELECT * FROM emp ORDER BY sal DESC;

SELECT * FROM emp ORDER BY deptno ASC;

SELECT * FROM emp ORDER BY hire;
-- default sort = ASC.

SELECT * FROM emp ORDER BY deptno ASC, job ASC;
-- sort on multiple columns.

SELECT * FROM emp ORDER BY deptno DESC, job, ename;
-- sort on deptno (desc), job (asc), ename (asc)
```

```
-- display top 3 emps as per sal.
SELECT * FROM emp ORDER BY sal DESC LIMIT 3;

-- display emp whose name is last in alphabetically.
SELECT * FROM emp ORDER BY ename DESC LIMIT 1;

SELECT * FROM emp ORDER BY comm;

-- display emp with lowest sal.
SELECT * FROM emp ORDER BY sal LIMIT 1;

-- display emp with third lowest sal.
SELECT * FROM emp ORDER BY sal LIMIT 2,1;

-- display emp with second highest sal.
```

```
SELECT * FROM emp ORDER BY sal DESC;

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

-- sort emp on da=sal*0.5
SELECT ename, sal, sal*0.5 da FROM emp;

SELECT ename, sal, sal*0.5 da FROM emp
ORDER BY sal*0.5;
-- order by expr

SELECT ename, sal, sal*0.5 da FROM emp
ORDER BY da;
-- order by column alias

SELECT ename, sal, sal*0.5 da FROM emp
ORDER BY 3;
-- order by column numer (in projection)
```

```
-- display emps of dept 30
SELECT * FROM emp WHERE deptno = 30;

-- display all emps with sal > 2000.0
SELECT * FROM emp WHERE sal > 2000.0;

-- display all ANALYST.
SELECT * FROM emp WHERE job = 'ANALYST';

-- display all emps not in dept 30.
SELECT * FROM emp WHERE deptno != 30;
SELECT * FROM emp WHERE deptno <> 30;
```

```
-- display emps in sal range from 1000 to 2000.
-- sal >= 1000 && sal <= 2000
SELECT * FROM emp WHERE sal >= 1000 AND sal <= 2000;

-- display ANALYSTs and MANAGERs.
-- job == 'ANALYST' || job == 'MANAGER';
SELECT * FROM emp WHERE job = 'ANALYST' OR job = 'MANAGER';

-- display emps who are not salesman.
SELECT * FROM emp WHERE job != 'SALESMAN';
SELECT * FROM emp WHERE job <> 'SALESMAN';

-- !(condition)
SELECT * FROM emp WHERE NOT job = 'SALESMAN';
```

```
-- display all emps hired in 1982.
-- '1-1-1982' to '31-12-1982'
SELECT * FROM emp WHERE hire >= '1982-01-01' AND hire <= '1982-12-31';
```

```
-- display all emps whose comm is null.
SELECT * FROM emp WHERE comm = NULL;
-- relational operators cannot be used with NULL.
-- NULL is used with special operators.

SELECT * FROM emp WHERE comm IS NULL;

SELECT * FROM emp WHERE comm <=> NULL;

-- display all emps whose comm is not null.
SELECT * FROM emp WHERE comm IS NOT NULL;

SELECT * FROM emp WHERE NOT (comm IS NULL);
```