Agenda

- Non Standard Joins
- Security (DCL)
- Transactions
- Row,Table & Pessimistic Locking

Non Standard Joins

```
-- display ename and dname.
SELECT e.ename, d.dname FROM emps e
INNER JOIN depts d ON e.deptno = d.deptno;

SELECT e.ename, d.dname FROM emps e
JOIN depts d ON e.deptno = d.deptno;
-- by default join is INNER (in MySQL).

SELECT e.ename, d.dname FROM emps e
CROSS JOIN depts d ON e.deptno = d.deptno;
-- In MySQL, we can apply condition on CROSS JOIN

SELECT e.ename, d.dname FROM emps e
CROSS JOIN depts d WHERE e.deptno = d.deptno;
-- You may use WHERE clause with CROSS JOIN
-- However choose INNER JOIN if applicable.

SELECT e.ename, d.dname FROM emps e, depts d
WHERE e.deptno = d.deptno;
-- Join without JOIN keyword is old-style join.

SELECT e.ename, d.dname FROM emps e
INNER JOIN depts d USING (deptno);
-- joined columns from both tables have SAME name
-- the condition can be given using USING keyword
-- USING (colname) --> t1.colname = t2.colname;
-- This is always eqi-join.
-- This works only in MySQL.

SELECT e.ename, d.dname FROM emps e
NATURAL JOIN depts d;
-- num of joined columns = 1 (same name)
-- NATURAL JOIN = Implicit Join Condition
-- Equality Check of Columns with Same Name in Both tables.
-- In this example
-- NATURAL JOIN: ON e.deptno = d.deptno.

-- display all possible depts for Amit & Nilesh.
SELECT e.ename, d.dname FROM emps e
CROSS JOIN depts d WHERE e.ename IN ('AMIT', 'NILESH');
```

DCL

- CREATE USER
  - % -> any client machine on network
  - ''@'localhost' -> local machine.
- GRANT
  - To provide permisssion to user on database and table
- REVOKE
  - To remove the permission from user from database and table
-

```
-- To cretae user
CREATE USER 'mgr'@'%' IDENTIFIED BY 'mgr';
CREATE USER 'teamlead'@'localhost' IDENTIFIED BY 'teamlead';
CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1';
CREATE USER 'dev2'@'localhost' IDENTIFIED BY 'dev2';

--to change the prompt name in mysql shell
PROMPT \u>
```

```
-- FROM root
GRANT ALL PRIVILEGES ON classwork_db.* TO 'mgr'@'%' WITH GRANT OPTIONS;
root>SHOW GRANTS FOR mgr;
root>SHOW GRANTS FOR teamlead;

-- FROM mgr
mgr> GRANT SELECT,INSERT,UPDATE,DELETE ON classwork.emp TO 'teamlead'@'localhost';
mgr> GRANT SELECT,INSERT, UPDATE, DELETE ON classwork.dept TO
'teamlead'@'localhost';
mgr> GRANT INSERT, SELECT ON classwork.books TO 'teamlead'@'localhost';
mgr> GRANT SELECT ON classwork.emp TO 'dev1'@'localhost';
mgr> REVOKE DELETE ON classwork.dept FROM 'teamlead'@'localhost';

-- FROM teamlead
teamlead> SHOW DATABASES;
teamlead> USE classwork_db;
teamlead> SHOW TABLES;
teamlead> DELETE FROM books; -- error: Access denied
```

Transations

- In MySQL, by default each DML operation is executed as a transaction with the single query and it is auto-committed.
- In MySQL, for transaction TCL commands are used. They are -
  - START TRANSACTION
    - to explicitly start transaction

- COMMIT
  - to commit/save the transaction
- ROLLBACK
  - to dicaard the changes done / undo.
- SAVEPOINT
  - to save the current state of transation before final commits.
  - multiple save points can be created.
  - we can rollback to any save points,however the rest all changes after the taht save point will be discarded.
  - we cannot commit to specific savepoint, only whole transaction gets commited

```
SELECT @@autocommit;
SET @@autocommit=0;
SELECT @@autocommit;
SET @@autocommit=1;

CREATE TABLE accounts(id INT PRIMARY KEY, type CHAR(20),balance DECIMAL(9,2));
INSERT INTO accounts VALUES
(1, 'Saving', 20000.00),
(2, 'Current', 60000.00),
(3, 'Saving', 5000.00),
(4, 'Saving', 3000.00),
(5, 'Current', 50000.00),
(6, 'Saving', 10000.00);
SELECT * FROM accounts;

-- use the mgr and teamlead for the transaction on this table

START TRANSACTION

SAVEPOINT s1;

SAVEPOINT s2;

ROLLBACK To s1;

ROLLBACK;

COMMIT;
```

- When an user is in a transation, changes done by the user are saved in a temp table. These changes are visible to that user.
- However this temp table is not accessible/visible to other users and hence changes under progress in a transaction are not visible to other users.
- When an user is in a transaction, changes committed by other users are not visible to him. Because he is dealing with temp data.

- If user is inside a transaction and once again starts transaction before rollback or commiting previous transaction then in this case the previous transactions is autocommited.
- Mysql Supports ACID Transaction
    - A -> Atomicity
        - All DML queries in transaction will be successful or failed/discarded. Partial transaction are never committed.
    - C -> Consistency
        - At the end of transaction same state is visible to all the users.
    - I -> Isolation
        - Each transaction is isolated from each other.
        - All transactions are added in a transaction queue at server side and process sequentially.
    - D -> Durability
        - At the end of transaction, final state is always saved (on server side)

Row,Table & Pessimistic Locking

- ROW LOCKING
    - When deleted or updated the row gets locked.
- Pessimistic Locking

```
SELECT * FROM accounts WHERE id = 2 FOR UPDATE;
```

- TABLE LOCKING
    - When deleted or updated the table gets locked if primary key does not exists