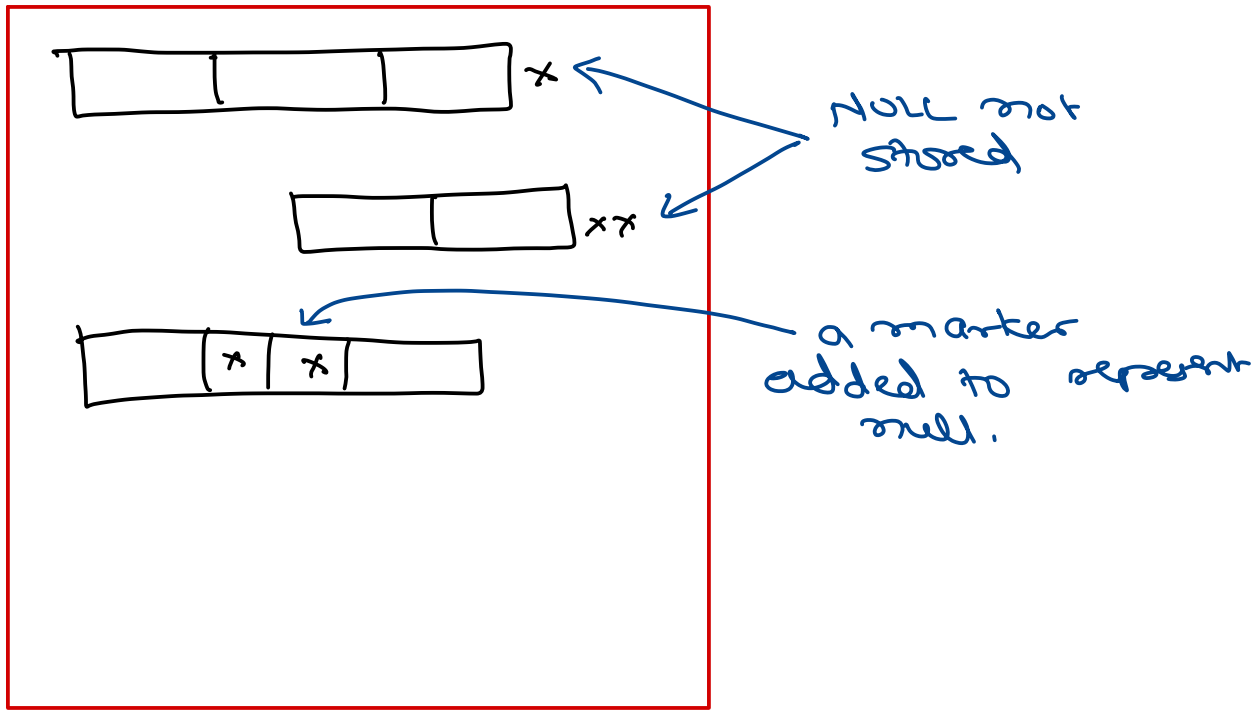
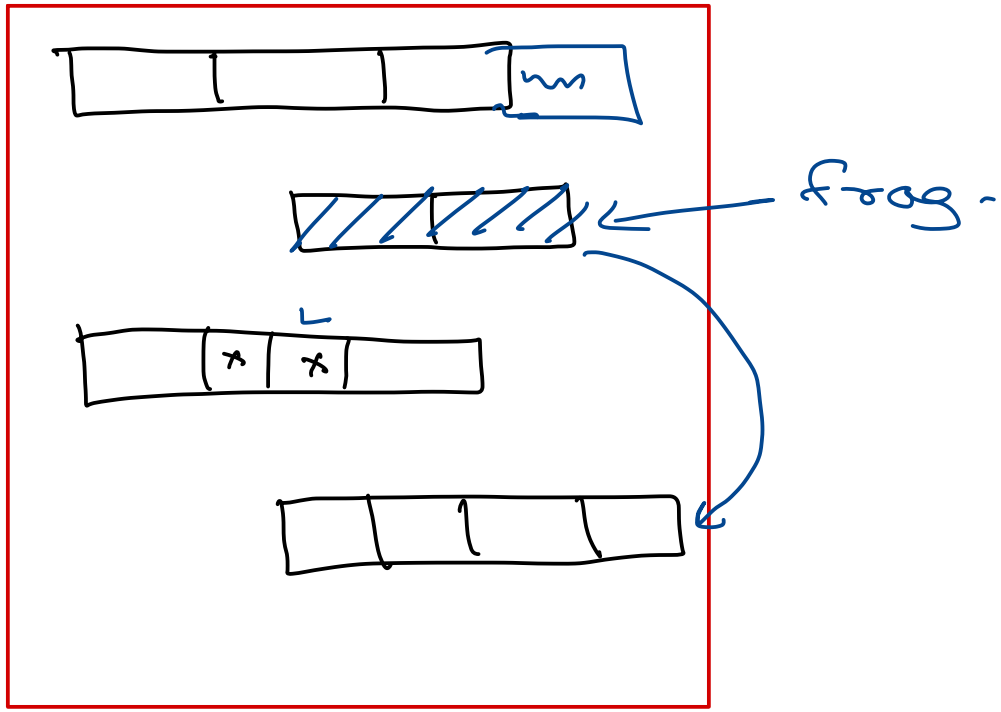

# MySQL RDBMS

Trainer: Mr. Nilesh Ghule

# NULL



# Transaction

- Transaction is set of DML queries executed as a single unit.
- Transaction examples
  - accounts table [id, type, balance]
  - UPDATE accounts SET balance=balance-1000 WHERE id = 1;
  - UPDATE accounts SET balance=balance+1000 WHERE id = 2;
- RDBMS transaction have ACID properties.
  - Atomicity
    - All queries are executed as a single unit. If any query is failed, other queries are discarded.
  - Consistency
    - When transaction is completed, all clients see the same data.
  - Isolation
    - Multiple transactions (by same or multiple clients) are processed concurrently.
  - Durable
    - When transaction is completed, all data is saved on disk.

# Transaction

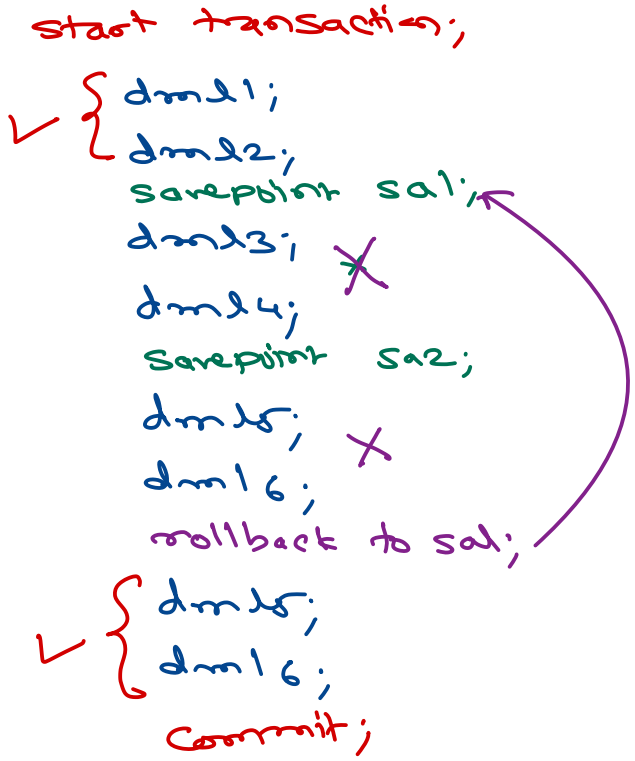
- Transaction management
  - START TRANSACTION;
  - …
  - COMMIT WORK;
  - START TRANSACTION;
  - …
  - ROLLBACK WORK;

WORK keyword optional.

- In MySQL autocommit variable is by default 1. So each DML command is auto-committed into database. → each query is one tx – auto commited.
  - SELECT @@autocommit;
- Changing autocommit to 0, will create new transaction immediately after current transaction is completed. This setting can be made permanent in config file.
  - SET autocommit=0;

# Transaction

- Save-point is state of database tables (data) at the moment (within a transaction).
- It is advised to create save-points at end of each logical section of work.
- Database user may choose to rollback to any of the save-point.
- Transaction management with Save-points
  - START TRANSACTION;
  - …
  - SAVEPOINT sa1;
  - …
  - SAVEPOINT sa2;
  - …
  - ROLLBACK TO sa1;
  - …
  - COMMIT; // or ROLLBACK
- Commit always commit the whole transaction.
- ROLLBACK or COMMIT clears all save-points.



# Transaction

- Transaction is set of DML statements.
- If any DDL statement is executed, current transaction is automatically committed.
- Any power failure, system or network failure automatically rollback current state.
- Transactions are isolated from each other and are consistent.

# Row locking

- When an user update or delete a row (within a transaction), that row is locked and becomes read-only for other users.
- The other users see old row values, until transaction is committed by first user.
- If other users try to modify or delete such locked row, their transaction processing is blocked until row is unlocked.
- Other users can INSERT into that table. Also they can UPDATE or DELETE other rows.
- The locks are automatically released when COMMIT/ROLLBACK is done by the user.
- This whole process is done automatically in MySQL. It is called as "OPTIMISTIC LOCKING".

# Row locking

- Manually locking the row in advanced before issuing UPDATE or DELETE is known as "PESSIMISTIC LOCKING".
- This is done by appending FOR UPDATE to the SELECT query.
- It will lock all selected rows, until transaction is committed or rollbacked.
- If these rows are already locked by another users, the SELECT operation is blocked until rows lock is released.

- By default MySQL does table locking. Row locking is possible only when table is indexed on the column.


# Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>