MySQL RDBMS

Trainer: Mr. Nilesh Ghule

MySQL Triggers

- Triggers are supported by all standard RDBMS like Oracle, MySQL, etc.
- Triggers are not supported by WEAK RDBMS like MS-Access, SQLite.
- Triggers are not called by client's directly, so they don't have args & return value.
- Trigger execution is caused by DML operations on database.
  - BEFORE/AFTER INSERT, BEFORE/AFTER UPDATE, BEFORE/AFTER DELETE.
- Like SP/FN, Triggers may contain SQL statements with programming constructs. They may also call other SP or FN.
- However COMMIT/ROLLBACK is not allowed in triggers. They are executed in same transaction in which DML query is executed.

CREATE TRIGGER

```
CREATE TRIGGER trig_name
AFTER|BEFORE dml_op ON table
FOR EACH ROW
BEGIN
   body;
   -- use OLD & NEW keywords
   -- to access old/new rows.
   -- INSERT triggers - NEW rows.
   -- DELETE triggers - OLD rows.
END;
```

UPDATE triggers

SHOW TRIGGERS

```
SHOW TRIGGERS FROM db_name;
```

DROP TRIGGER

```
DROP TRIGGER trig_name;
```

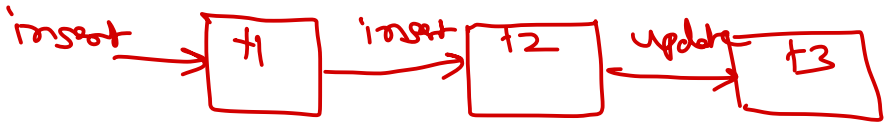
MySQL Triggers

- Applications of triggers:
  - Maintain logs of DML operations (Audit Trails).
  - Data cleansing before insert or update data into table. (Modify NEW value).
  - Copying each record AFTER INSERT into another table to maintain "Shadow table".
  - Copying each record AFTER DELETE into another table to maintain "History table".
  - Auto operations of related tables using cascading triggers.

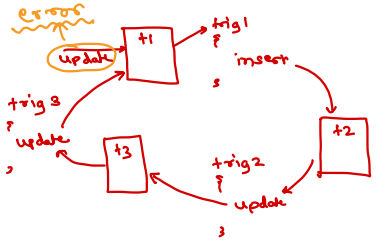
CHECK constraint for checking value.

SET NEW.ename = UPPER(OLD.ename);

emp → sal_history



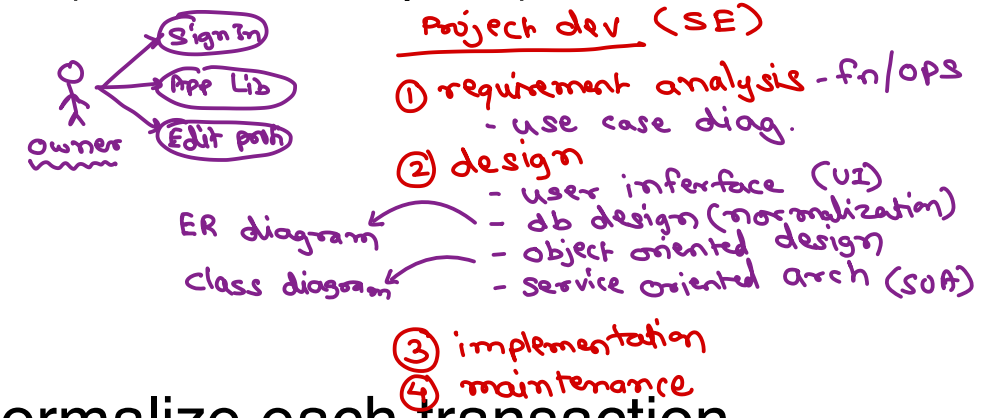
- Cascading triggers
  - One trigger causes execution of 2nd trigger, 2nd trigger causes execution of 3rd trigger and so on.
  - In MySQL, there is no upper limit on number of levels of cascading.
  - This is helpful in complicated business processes.



- Mutating table error
  - If cascading trigger causes one of the earlier trigger to re-execute, "mutating table" error is raised.
  - This prevents infinite loop and also rollback the current transaction.

Normalization

- Concept of table design: Table, Structure, Data Types, Width, Constraints, Relations.
- Goals:
  - Efficient table structure.
  - Avoid data redundancy i.e. unnecessary duplication of data (to save disk space).
  - Reduce problems of insert, update & delete.
- Done from input perspective.
- Based on user requirements.
- Part of software design phase.
- View entire appln on per transaction basis & then normalize each transaction separately.
- Transaction Examples:
  - Banking, Rail Reservation, Online Shopping.



Normalization

- For given transaction make list of all the fields.
- Strive for atomicity.
- Get general description of all field properties.
- For all practical purposes we can have a single table with all the columns. Give meaningful names to the table.
- Assign datatypes and widths to all columns on the basis of general desc of fields properties.
- Remove computed columns.
- Assign primary key to the table.
- At this stage data is in un-normalized form.
- UNF is starting point of normalization.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>