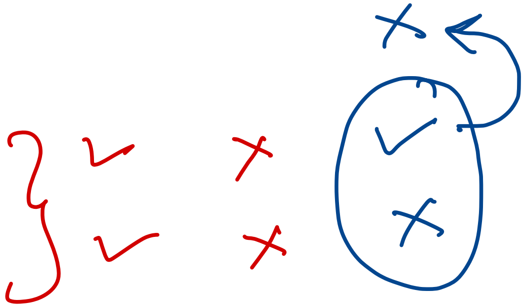
MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

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;

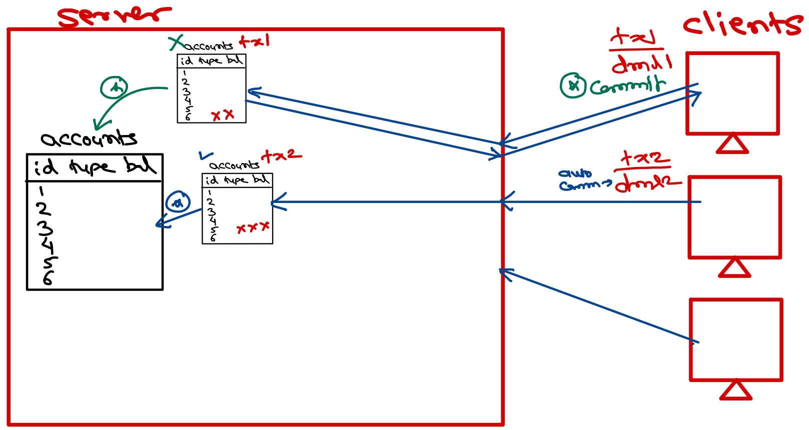


- RDBMS transaction have ACID properties.
  - Atomicity
    - All queries are executed as a single unit. If any query is failed, other queries are discarded.
  - Consistency
    - When transaction is completed, all clients see the same data.
  - Isolation
    - Multiple transactions (by same or multiple 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;



- In MySQL autocommit variable is by default 1. So each DML command is auto-committed into database.
  - SELECT @@autocommit; → 1
- 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

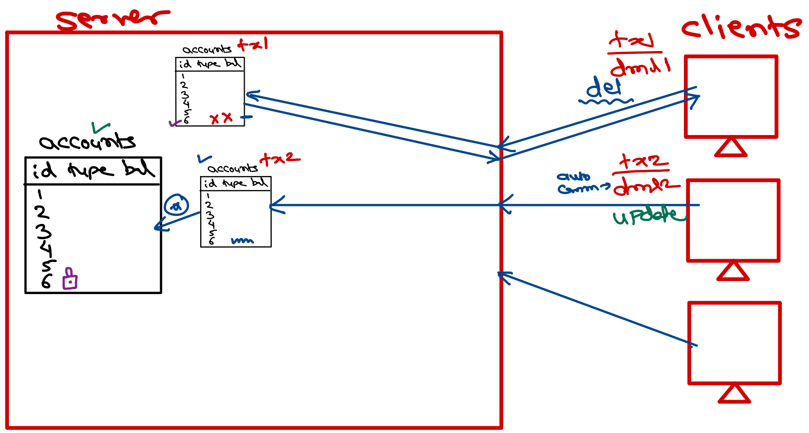
- Save-point is state of database tables (data) at the moment (within a transaction).
- It is advised to create save-points at end of each logical section of work.
- Database user may choose to rollback to any of the save-point.
- Transaction management with Save-points
  - START TRANSACTION;
  - …
  - 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. or timeout occurs.
- This whole process is done automatically in MySQL. It is called as "OPTIMISTIC LOCKING".



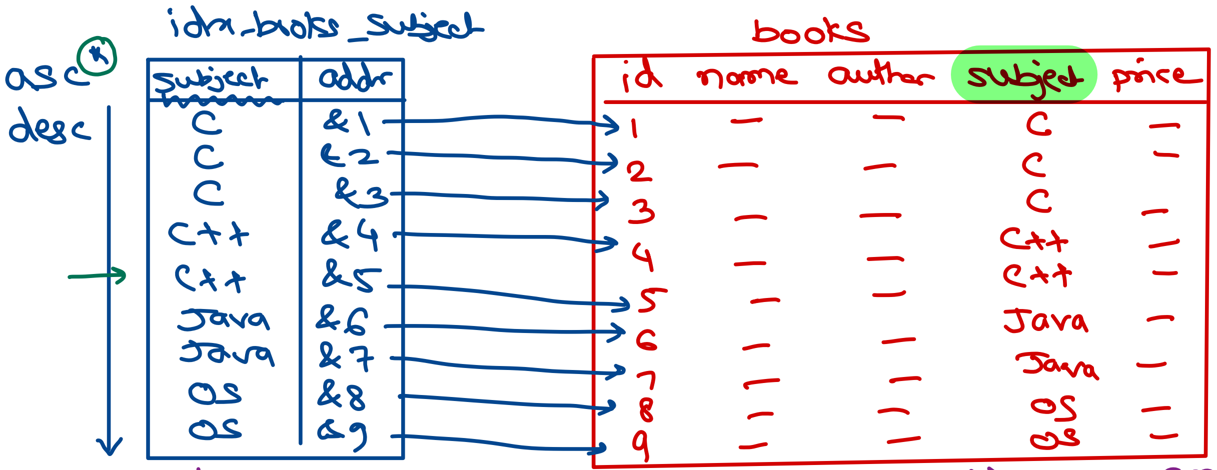
Row locking

- 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. or timeout.
- 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. (pri key)

Index

- Index enable faster searching in tables by indexed columns.
  - CREATE INDEX idx_name ON table(column);
- One table can have multiple indexes on different columns/order.
- Typically indexes are stored as some data structure (like BTREE or HASH) on disk.
- Indexes are updated during DML operations. So DML operation are slower on indexed tables.



idx_books_subject

subject	addr
C	&1
C	&2
C	&3
C++	&4
C++	&5
Java	&6
Java	&7
OS	&8
OS	&9

books

id	name	author	subject	price
1	—	—	C	—
2	—	—	C	—
3	—	—	C	—
4	—	—	C++	—
5	—	—	C++	—
6	—	—	Java	—
7	—	—	Java	—
8	—	—	OS	—
9	—	—	OS	—

↳ internally stored on server hard disk. → RDBMS specific.
In MySQL indexes are stored as BTree or Hash.

Index

- Index can be ASC or DESC.
  - It cause storage of key values in respective order (MySQL 8.x onwards).
  - ASC/DESC index is used by optimizer on ORDER BY queries.
- There are four types of indexes:
  - Simple index
    - CREATE INDEX idx_name ON table(column [ASCIDESC]);
  - Unique index
    - CREATE UNIQUE INDEX idx_name ON table(column [ASCIDESC]);
    - Doesn’t allow duplicate values.
  - Composite index
    - CREATE INDEX idx_name ON table(column1 [ASCIDESC], column2 [ASCIDESC]);
    - Composite index can also be unique. Do not allow duplicate combination of columns.
  - Clustered index
    - PRIMARY index automatically created on Primary key for row lookup.
    - If primary key is not available, hidden index is created on synthetic column.
    - It is maintained in tabular form and its reference is used in other indexes.

Index

- Indexes should be created on shorter (INT, CHAR, …) columns to save disk space.
- Few RDBMS do not allow indexes on external columns i.e. TEXT, BLOB.
- MySQL support indexing on TEXT/BLOB up to n characters.
  - CREATE TABLE test (blob_col BLOB, …, INDEX(blob_col(10)));
- To list all indexes on table:
  - SHOW INDEXES FROM table;
- To drop an index:
  - DROP INDEX idx_name ON table;
- When table is dropped, all indexes are automatically dropped.
- Indexes should not be created on the columns not used frequent search, ordering or grouping operations.
- Columns in join operation should be indexed for better performance.

Query performance

- Few RDBMS features ensure better query performance.
  - Index speed up execution of SELECT queries (search operations).
  - Correlated sub-queries execute faster.
- Query performance can observed using EXPLAIN statement.
  - EXPLAIN FORMAT=JSON SELECT …;
- EXPLAIN statement shows
  - Query cost (Lower is the cost, faster is the query execution).
  - Execution plan (Algorithm used to execute query e.g. loop, semi-join, materialization, etc).
- Optimizations can be enabled or disabled by optimizer_switch system variable.
  - SELECT @@optimizer_switch;
  - SET @@optimizer_switch=‘materialization=off’;

Constraints

- Constraints are restrictions imposed on columns.
- There are five constraints
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
- Few constraints can be applied at either column level or table level. Few constraints can be applied on both.
- Optionally constraint names can be mentioned while creating the constraint. If not given, it is auto-generated.
- Each DML operation check the constraints before manipulating the values. If any constraint is violated, error is raised.

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>