Transactions

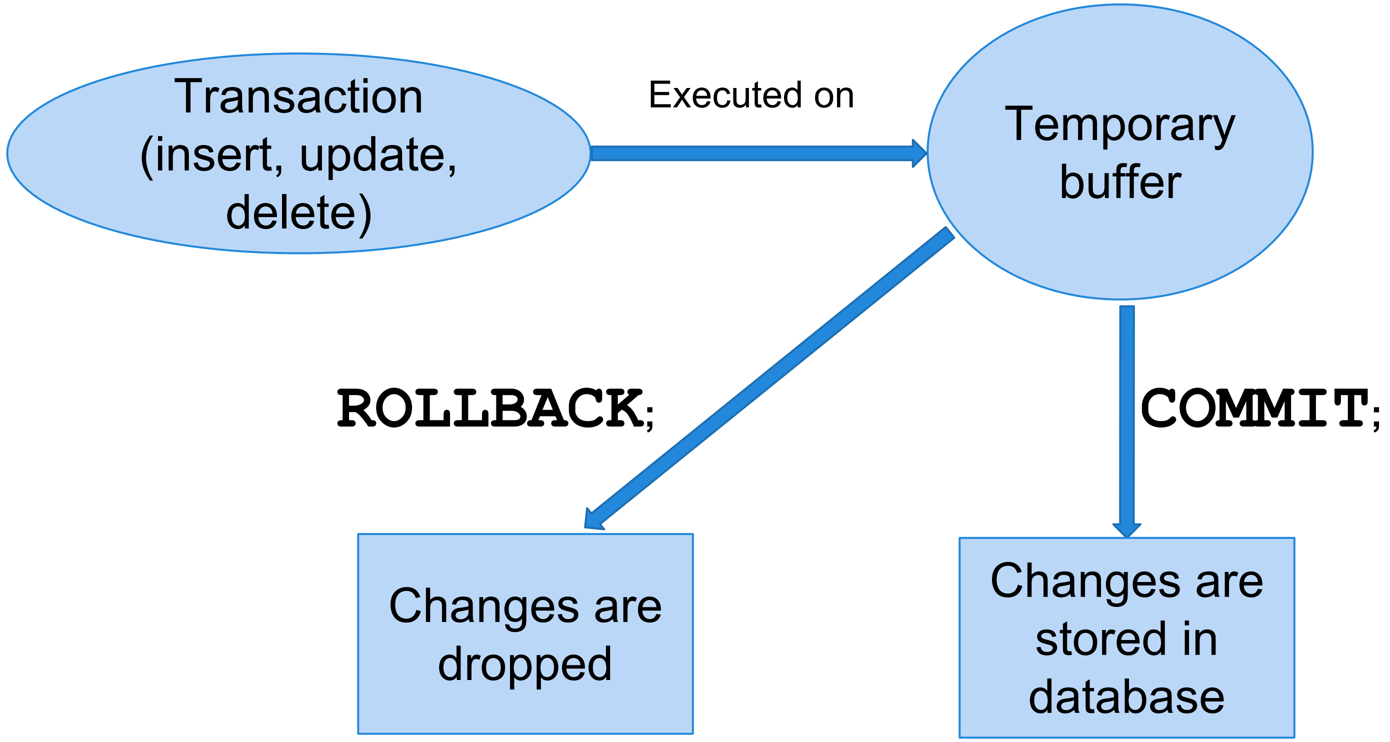
J. Porubän, M. Biňas, M. Nosáľ,
D. Lakatoš (c) 2011 - 2018

Transaction

- sequence of one or more SQL statements to be treated as a one atomic unit
- carried out in isolation
  - no operation may be performed during this transaction to affect it
- at time of system failure either all changes are applied or no changes are applied at all
  - DB remains in a consistent state

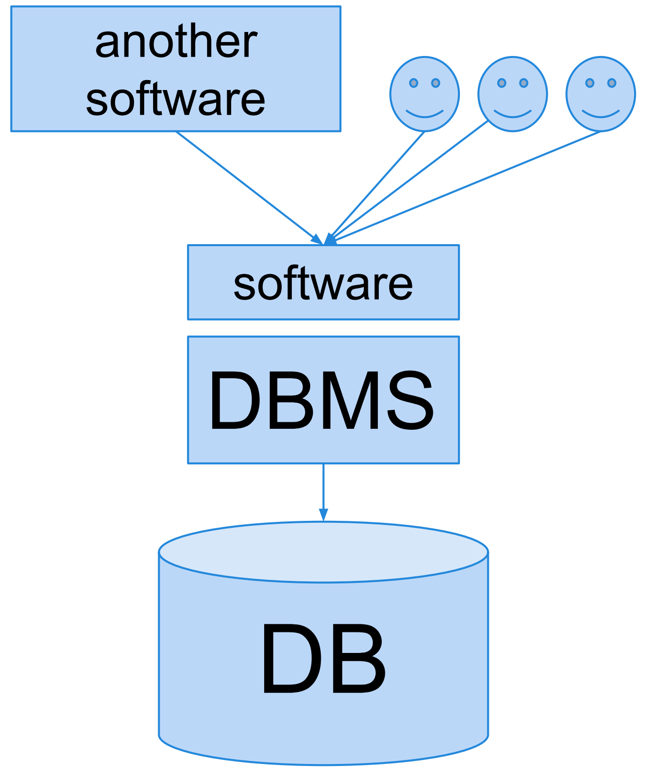
Transactional execution

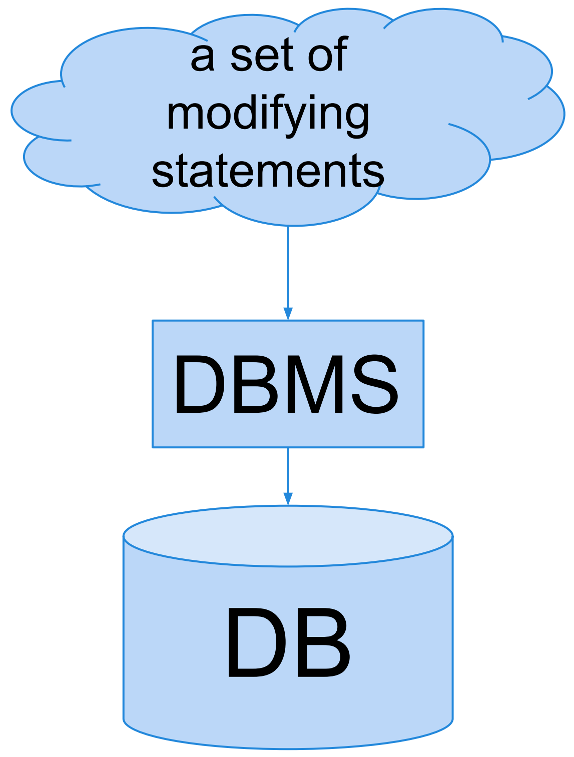
- Sequence of modifying statements executed as a whole - transaction
  - Either all changes are stored
  - Or none is stored


Transaction (insert, update, delete)
Executed on
Temporary buffer
ROLLBACK;
COMMIT;
Changes are dropped
Changes are stored in database

Motivation for transaction

- Simultaneous (parallel) access to the database
- Resistance to system failures





Goals of transactions

- parallel access to the database
  - executing a sequence of commands as if performed in isolation
  - allows simultaneous execution whenever possible
- resistance to system failures
  - guarantee that either everything is executed or nothing is executed, regardless of possible system failure

Parallel access

- database system should support parallel access for multiple users
- while maintaining the integrity and consistency of data
- during paralell access different types of inconsistency can occur:
  - at the attribute level
  - at the record (row) level
  - at the table level
  - at the level of multiple statements

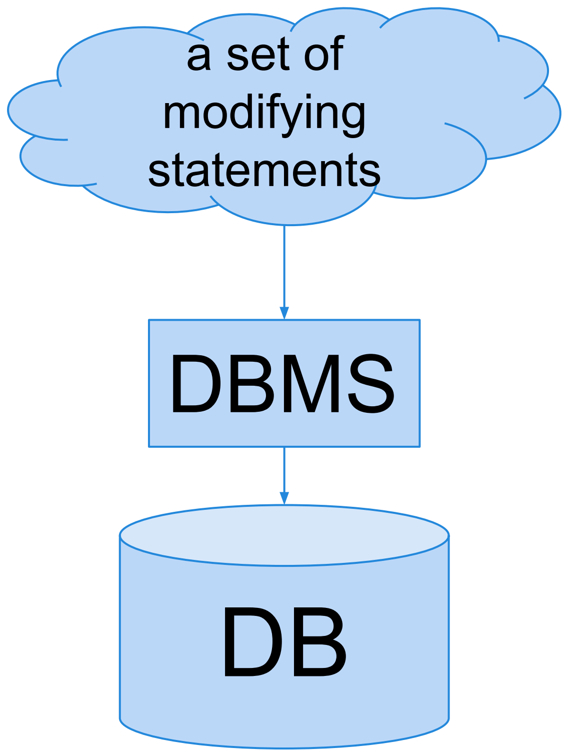
Example

```
UPDATE   account
SET      balance = balance - 100
WHERE    id = 21;
UPDATE   account
SET      balance = balance + 100
WHERE    id = 22;
```

```
UPDATE   account
SET      balance = balance * 1.02
WHERE    type = 'common';
```

Resistance to system failures

- the system should ensure data consistency even in case of system failure
- failure during a series of commands (database initialization)
- during data migration in DB
- while the updates are in a memore (not stored to hard drive, yet)



Example

```
UPDATE  account
SET     balance = balance - 100
WHERE   id = 21;


UPDATE  account
SET     balance = balance + 100
WHERE   id = 22;
```

Transactions in SQL

- starts automatically with the first statement
- using the **`commit`** keyword command will finish the transaction and start a new one
- the current transaction ends at the end of the session
- `autocommit` – mode: each statement is in it’s own transaction
- changes done in current transaction can be canceled by the `rollback` keyword command

Example

```
START TRANSACTION;
INSERT INTO student
VALUES (10, 'Janko', 'Hraško', 4);
COMMIT;

START TRANSACTION;
UPDATE  student
SET     name = 'Jozko'
WHERE   id = 10;
ROLLBACK;
```

ACID

- **atomicity**
  - all or nothing (`rollback` allows to cancel all changes made in transaction)
- **consistency**
  - for each transaction, it can be assumed that all DB restrictions are met before it starts and must ensure that are met after it is completed

ACID

- **isolation**
  - serializability - the result of operations corresponds to some sequential execution of all transactions
  - 4 levels of isolation (Serializable, Repeatable Read, Read Committed, Read Uncommitted)
- **durability**
  - after commit, all changes must remain in the database

Questions?