Transactions

Relational Databases Basics

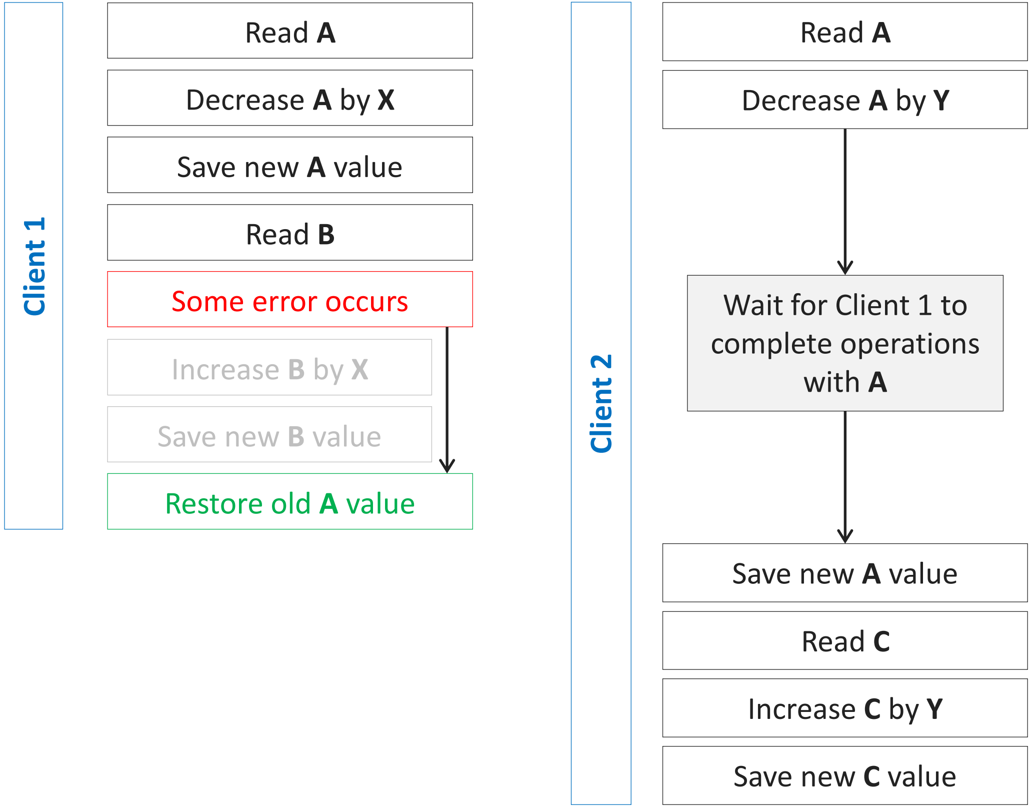
TRAINING CENTER

Transaction

Read and remember!

Transaction – a set of operations that forms an inseparable unit, i.e. either all operations are successfully completed, or all operations are cancelled (if at least one operation has failed).

Transaction logic



Client 1

Read **A**

Decrease **A** by **X**

Save new **A** value

Read **B**

Some error occurs

Increase **B** by **X**

Save new **B** value

Restore old **A** value

Client 2

Read **A**

Decrease **A** by **Y**

Wait for Client 1 to complete operations with **A**

Save new **A** value

Read **C**

Increase **C** by **Y**

Save new **C** value

The ACID principles

A **Atomicity:** no operations may be completed or cancelled “on their own”

C **Consistency:** final transaction results may not violate database consistency

I **Isolation:** transactions may not ruin each other’s work

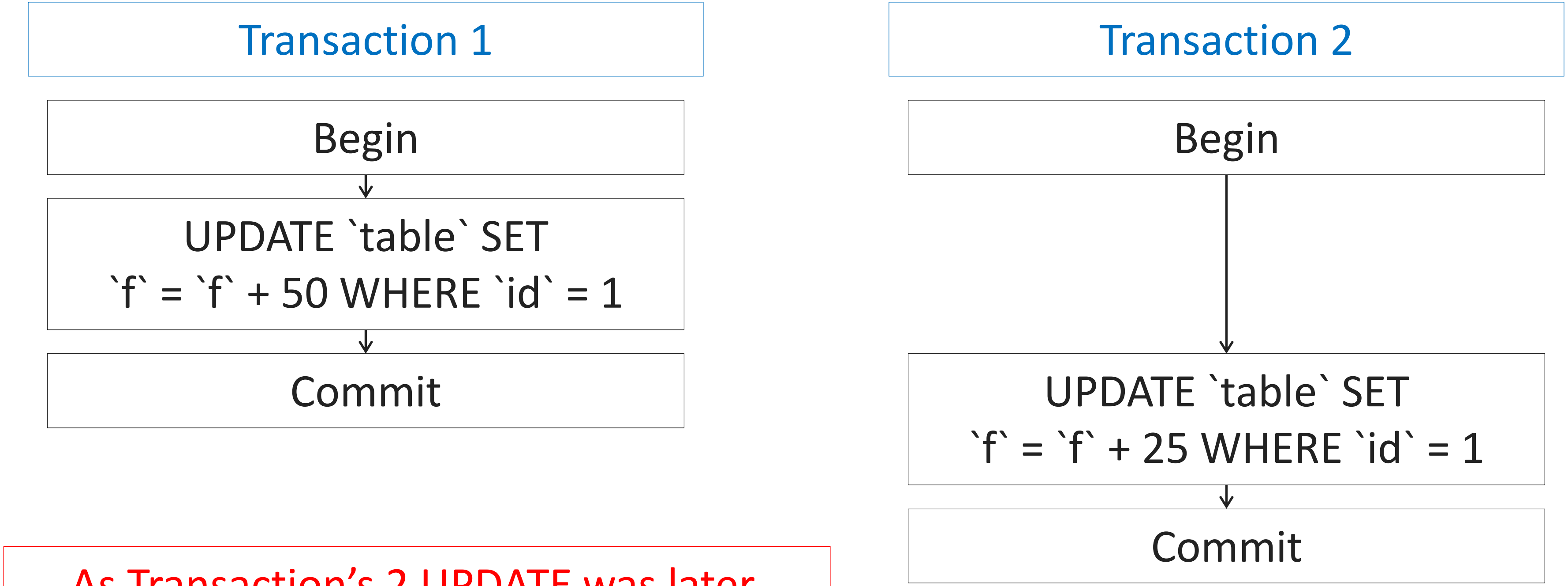
D **Durability:** once transaction is committed, nothing will “disappear”

Transaction isolation levels

Read and remember!

Transaction isolation level – a virtual value showing the degree of visibility of changes between transactions.

Potential problems: lost update



Transaction 1

Begin

UPDATE \`table\` SET \`f\` = \`f\` + 50 WHERE \`id\` = 1

Commit

Transaction 2

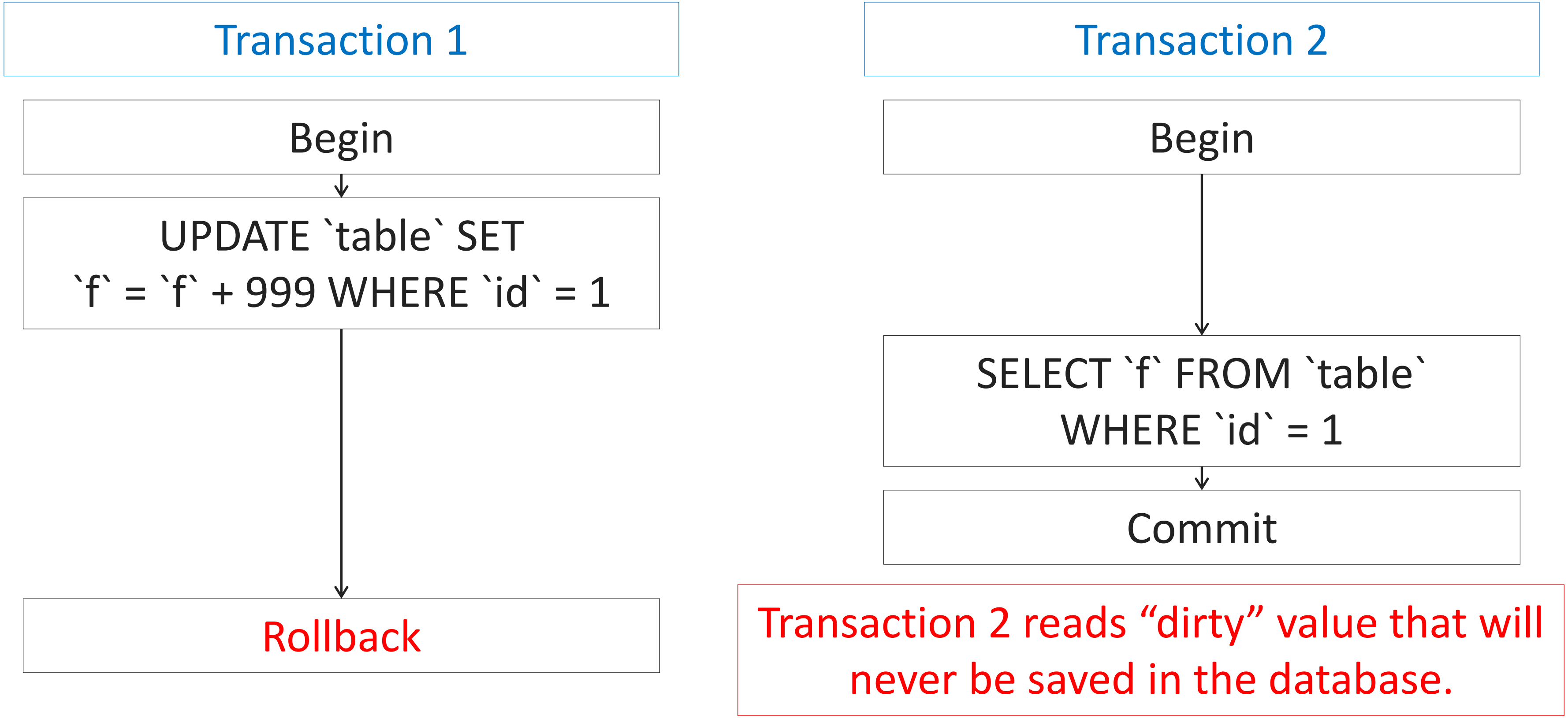
Begin

UPDATE \`table\` SET \`f\` = \`f\` + 25 WHERE \`id\` = 1

Commit

As Transaction's 2 UPDATE was later, changes from Transaction 1 are lost.

Potential problems: dirty read



Transaction 1

Begin

UPDATE `table` SET `f` = `f` + 999 WHERE `id` = 1

Rollback

Transaction 2

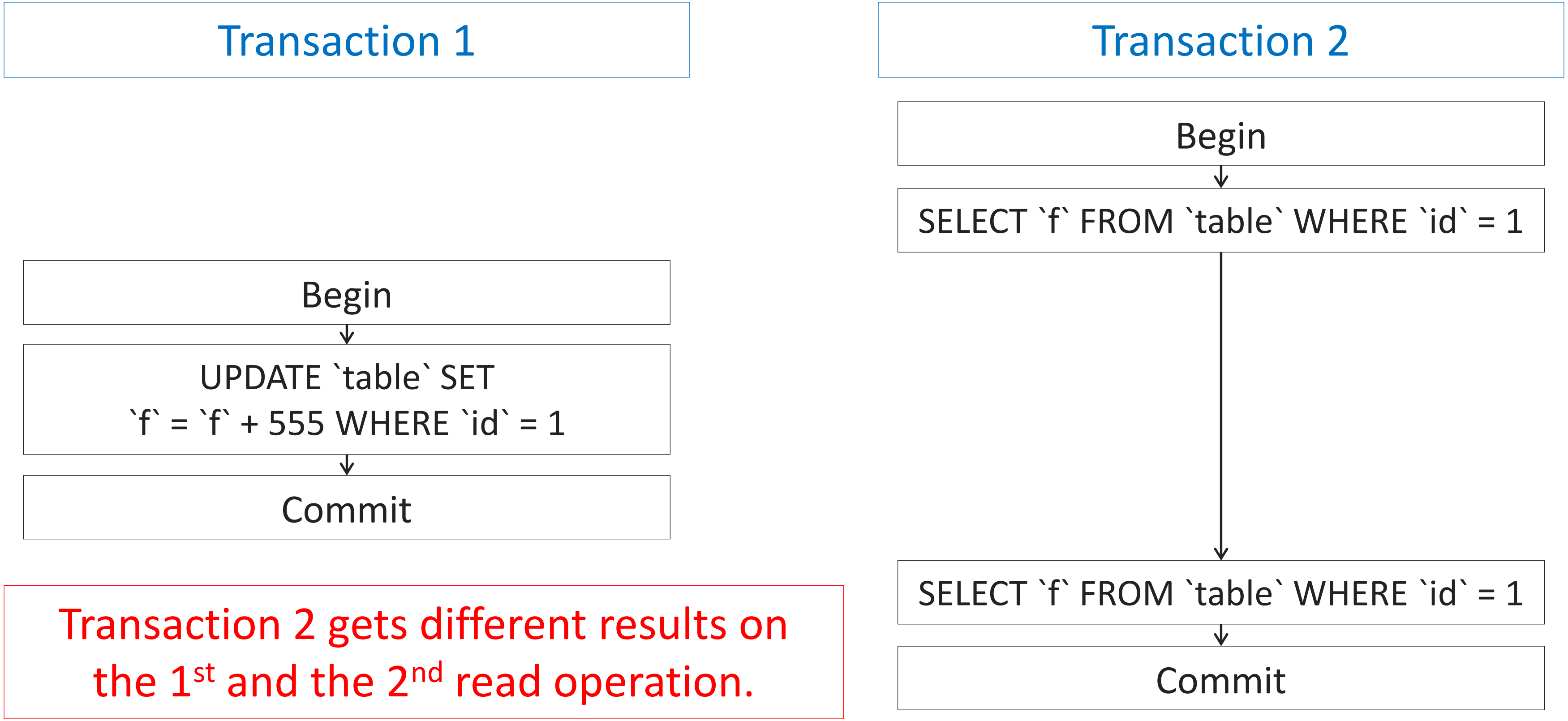
Begin

SELECT `f` FROM `table` WHERE `id` = 1

Commit

Transaction 2 reads “dirty” value that will never be saved in the database.

Potential problems: non-repeatable read



Transaction 1

Begin

UPDATE \`table\` SET \`f\` = \`f\` + 555 WHERE \`id\` = 1

Commit

Transaction 2

Begin

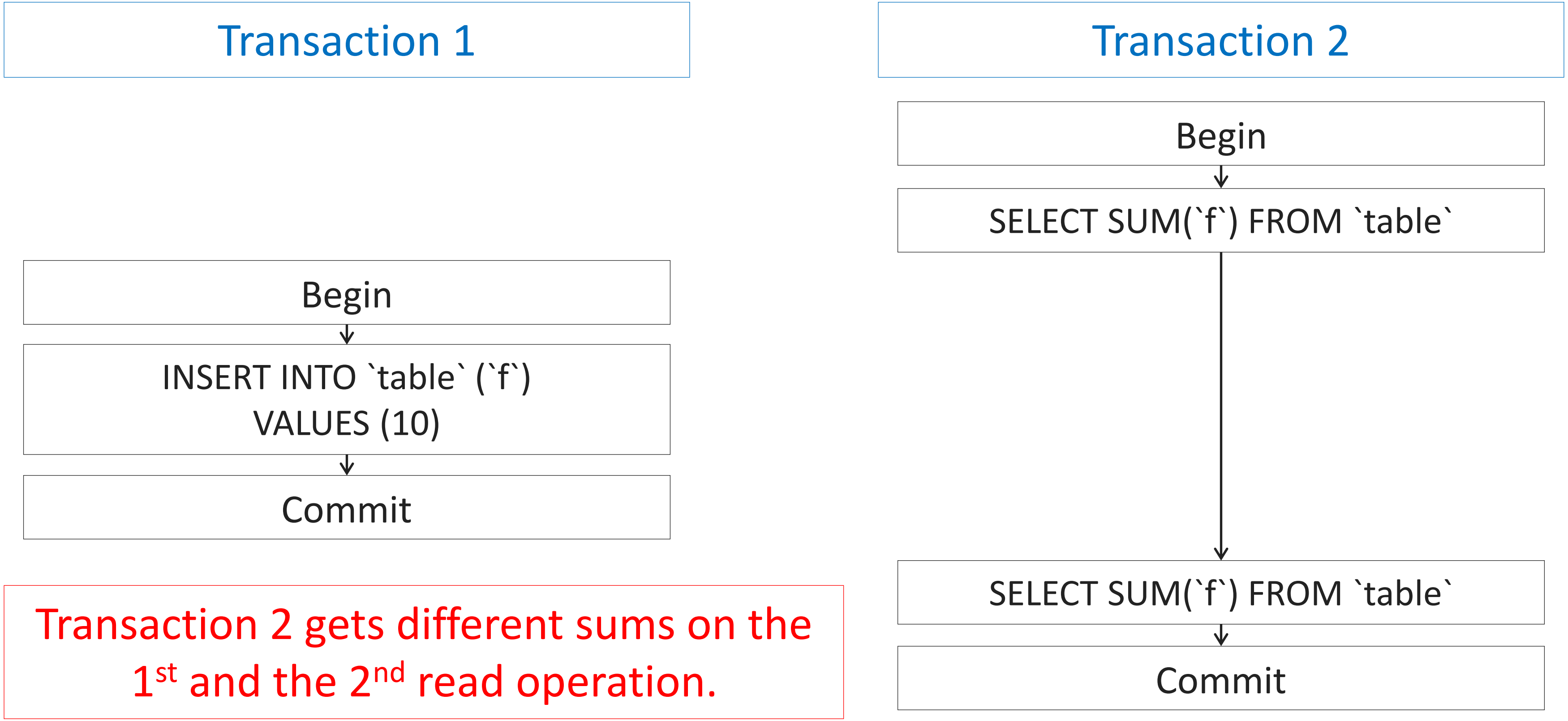
SELECT \`f\` FROM \`table\` WHERE \`id\` = 1

SELECT \`f\` FROM \`table\` WHERE \`id\` = 1

Commit

Transaction 2 gets different results on the 1st and the 2nd read operation.

Potential problems: phantom reads



Transaction 1

Begin

INSERT INTO `table` (`f`)
VALUES (10)

Commit

Transaction 2

Begin

SELECT SUM(`f`) FROM `table`

SELECT SUM(`f`) FROM `table`

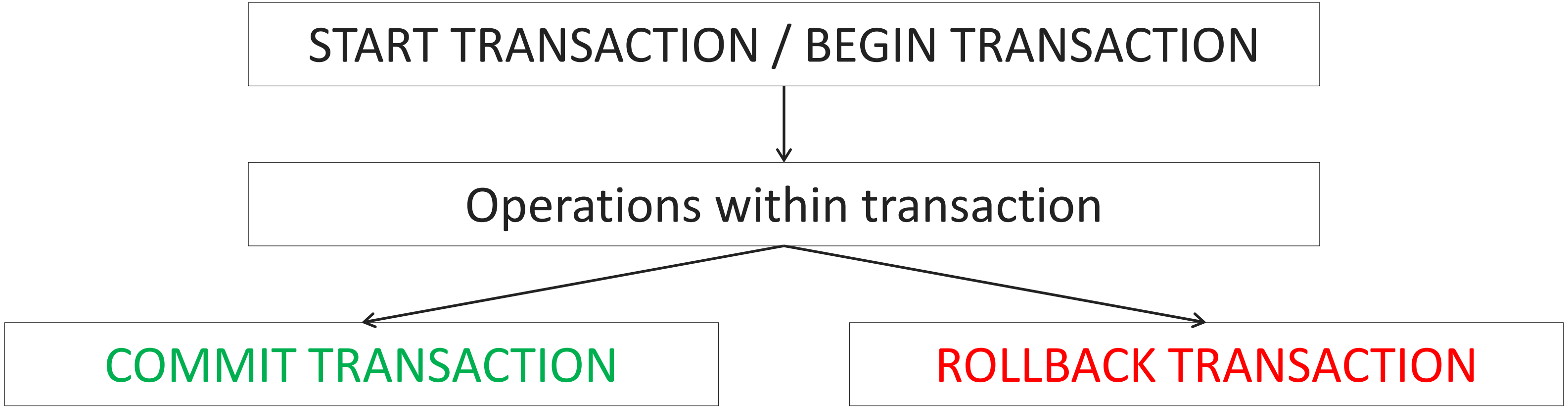
Commit

Transaction 2 gets different sums on the 1st and the 2nd read operation.

Transaction isolation levels hierarchy

	Lost update	**Dirty read**	**Non-repeatable read**	**Phantom reads**
Serializable	Protects	Protects	Protects	Protects
Snapshot	Protects	Protects	Protects	Doesn’t protect
Repeatable read	Protects	Protects	Protects	Doesn’t protect
Read committed	Protects	Protects	Doesn’t protect	Doesn’t protect
Read uncommitted	Protects	Doesn’t protect	Doesn’t protect	Doesn’t protect

Transaction management



START TRANSACTION / BEGIN TRANSACTION

Operations within transaction

COMMIT TRANSACTION

ROLLBACK TRANSACTION

```sql
START TRANSACTION;
DELETE FROM `user`
       WHERE `u_id` = 1000;
COMMIT;
```

Transaction management

Fast (maybe inaccurate) operation

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
START TRANSACTION;
SELECT YEAR(FROM_UNIXTIME(`f_upload_dt`)) AS `Y`,
       MONTH(FROM_UNIXTIME(`f_upload_dt`)) AS `M`,
       DAY(FROM_UNIXTIME(`f_upload_dt`)) AS `D`,
       COUNT(`f_id`) AS `files`
  FROM `file`
 GROUP BY `Y`, `M`, `D`;
COMMIT;
```

Slow (surely accurate) operation

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
SELECT `r_name`,
       COUNT(`ur_user`) as `users`
  FROM `role`
  JOIN `m2m_user_role`
    ON `r_id` = `ur_role`;
COMMIT;
```

Live demo in MySQL Workbench

Transactions

Relational Databases Basics