TRANSACTIONS

CS-A1153 - Databases (Summer 2020)

LUKAS AHRENBERG

Aalto University
School of Science

TRANSACTIONS

- The problem transactions are addressing
- Atomicity, Consistency, Isolation, Durability : **ACID**
- Transactions in SQL
- Isolation levels in SQL

A transaction is a collection of one or more operations on the database that must be executed atomically; that is, either all operations are performed or none are.

U&W 1:24, 6:6

Aalto University
School of Science

SOME ISSUES

Service loss

What happens if the system goes down in the middle of a bank transfer?

Multiple users

What if user A has selected the same seat as user B at the same time?

SOME SOLUTIONS

Transactional properties, usually enforced by

Logging
: There should be an unambiguous record of what has happened

Concurrency control
: What can happen 'at the same time', and what can not

Deadlock resolution
: Stop circular dependencies where no task goes first

Can lead to complex problems

ACID

Atomicity

If there is a failure halfway through a transaction, the DB should not be able to end up in an unacceptable state

Consistency

A transaction can not violate constraints set on the database

Isolation

(Serializability) Two transactions should have the same effect as if they happened in isolation, one before the other

Durability

The effect of a translation can never be lost once it is complete

TRANSACTIONS IN SQL

- In SQL each statement is a transaction by itself
- A set of statements can be grouped to a transaction by using `START TRANSACTION` and ended by either `ROLLBACK` or `COMMIT`

```
START TRANSACTION
<statements>
COMMIT;
```

```
START TRANSACTION
<statements>
ROLLBACK;
```

Aalto University
School of Science

READ ONLY TRANSACTIONS

By default a transaction is read/write, meaning that it offers consistency for both reading and writing.

If it is known that the transaction will not make changes to the data, it can be declared read only:

```
SET TRANSACTION READ ONLY;
START TRANSACTION
...
;
```

This allows the DB system more concurrency and potentially better efficacy, but at the cost of retrieving potentially out-of-date information. (Crucially, however, not corrupted.)

ISOLATION LEVELS

The *isolation level* of a transaction specifies **what that particular transaction may see**.

SERIALIZABLE

No other transaction may write to the data fields this transaction is working with until it finishes

REPEATABLE READ

This transaction *can read committed data* by other transactions which may execute simultaneously and *repeated reads* within this transaction must be consistent

READ COMMITTED

This transaction *can read committed data* by other transactions which may execute simultaneously, but repeated reads not necessarily consistent

READ UNCOMMITTED

This transaction can read 'dirty' data, *not yet committed* by other transactions

REPEATED READS AND PHANTOM TUPLES

For repeatable reads, other transactions may make changes to tables read by the transaction. However, only in such a way that repeated reads within the transaction result in the same or a super set of the same tuples. Any extra tuples gotten by subsequent reads are called **phantom tuples.**

```
SET TRANSACTION
    ISOLATION LEVEL REPEATABLE READ;
```

Aalto University
School of Science

READ COMMITTED

The transaction may read different data depending on when it is executed.

```
SET TRANSACTION READ WRITE
    ISOLATION LEVEL READ COMMITTED;
```

UNCOMMITTED (A.K.A DIRTY) READING

This is sometimes called *dirty* reading, and result in **dirty data**. Data which is first written by transaction A, and read dirty by transaction B and used in some way, then rolled back by transaction A.

Might be OK depending on application. Movie-ticket reservation - possibly; Banking - nope.

```
SET TRANSACTION READ WRITE
    ISOLATION LEVEL READ UNCOMMITTED;
```

Aalto University
School of Science

ISOLATION LEVELS

Isolation Level	Dirty Reads	Non-repeatable Reads	Phantoms
Read Uncommitted	Allowed	Allowed	Allowed
Read Committed	Not Allowed	Allowed	Allowed
Repeatable Read	Not Allowed	Not Allowed	Allowed
Serializable	Not Allowed	Not Allowed	Not Allowed

- Default: Serializable
- Why change?
    - Speedup. Transaction spend less time waiting
- **At the price of potential data inconsistencies for the transaction**
    - A transaction level only has effect on one transaction, not other ones
    - For some applications this is acceptable