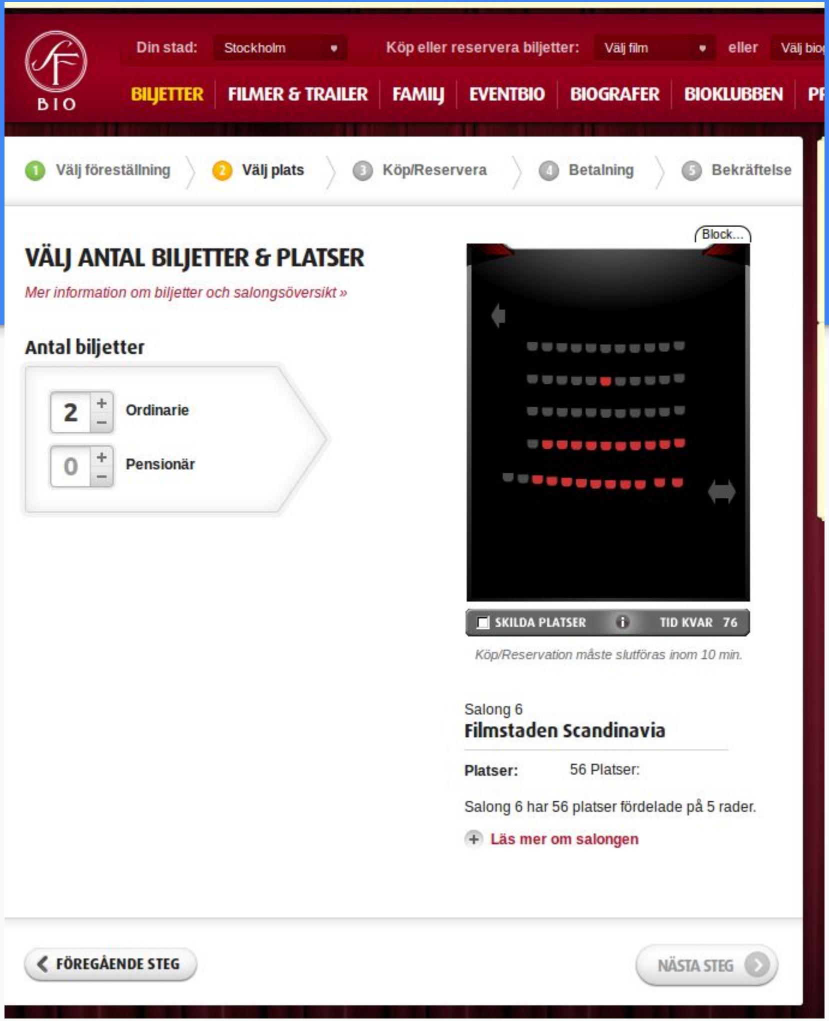
Transactions

Isolation

Concurrency means Trouble

Several processes manipulating the same data at the same time can lead to **inconsistencies**



VÄLJ ANTAL BILJETTER & PLATSER

Mer information om biljetter och salongsöversikt »

Antal biljetter

2	Ordinarie
0	Pensionär

Köp/Reservation måste slutföras inom 10 min.

Salong 6
Filmstaden Scandinavia

Platser: 56 Platser:

Salong 6 har 56 platser fördelade på 5 rader.

Läs mer om salongen

Transferring Money

Basic database operations (Insert, Update, Select) are atomic.

Sometimes we cannot write one single basic atomic operation

- Subtract 10$ from account X
- Add 10$ to account Y

What problems can occur? Power failure, account Y has some constraints (or doesn’t exist)

Transaction

- Sequence of one or more SQL operations treated as a unit
- Operations may be interleaved
- **Solution to concurrency and system failure.**
- ACID guarantee that database transactions are processed reliably.

ACID

Atomicity : requires that each transaction be "**all or nothing**": if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. **(rollback) , transfer is cancelled.**

Consistency : ensures that any transaction will bring the database from one valid state to another. (No money will be created or disappear)

Isolation : The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially

Durability :The durability property ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.

Read phenomena

Dirty Read -- **Read a uncommitted data** that is modified by another transaction**.** This value is not consistant as if the other transaction crashes (or rollback) the value is not valid anymore. There in no dirty read inside a transaction

Non-repeatable Read -- **Data changes between read and read**. A session can read a row in a transaction. Another session then changes the row (UPDATE or DELETE) and commits its transaction. If the first session subsequently re-reads the row in the same transaction, it will see the change.

Phantoms -- **Read operation does not affect new row inserted to DB.** A session can read a set of rows in a transaction that satisfies a search condition (which might be all rows). Another session then generates a row (INSERT) that satisfies the search condition and commits its transaction. If the first session subsequently repeats the search in the same transaction, it will see the new row.

Locks

Read lock

Write Lock

Range Lock !

Locks are set on rows and tables .

Isolation levels

1- Serializable (highest level, No interference) Requires read and write locks (acquired on selected data) to be released at the end of the transaction. Also range-locks must be acquired when a SELECT query uses a ranged WHERE clause (avoid the ***phantom reads*** phenomenon)

4- Repeatable reads **1) No dirty read, 2)a value of an item does not change after multiple read, but new items can be added/interfere meanwhile**. Read and write locks (acquired on selected data) until the end of the transaction. However, *range-locks* are not managed, so *phantom reads* can occur

3- Read committed **They don't perform dirty read. but *non-repeatable reads* phenomenon can occur in this isolation level if other transaction commit between reads**. it keeps write locks (acquired on selected data) until the end of the transaction, but read locks are released as soon as the SELECT operation is performed. As in the previous level, *range-locks* are not managed.

2- Read uncommitted (lowest level) **dirty reads are allowed**, so one transaction may see not-yet-committed changes made by other transactions.

Isolation summary

	Dirty Reads	Non-repeatable Reads	Phantoms
Read Uncommitted	Y	Y	Y
Read Committed	N	Y	Y
Repeatable Read	N	N	Y
Serializable	N	N	N

Anomaly - Dirty reads

```
/* Transaction 1 */
SELECT age FROM users WHERE id = 1;
/* will read 20 */

                    /* Transaction 2 */
                    UPDATE users SET age = 21 WHERE id = 1;
                    /* No commit here */

/* Transaction 1 */
SELECT age FROM users WHERE id = 1;
/* will read 21 */

                    ROLLBACK; /* lock-based DIRTY READ */
```

Anomaly - Non-repeatable reads

```
/* Transaction 1 */
SELECT * FROM users WHERE id = 1;

                        /* Transaction 2 */
                        UPDATE users SET age = 21 WHERE id = 1;
                        COMMIT; /* in multiversion concurrency
                          control, or lock-based READ COMMITTED */

/* Transaction 1 */
SELECT * FROM users WHERE id = 1;
COMMIT; /* lock-based REPEATABLE READ */
```

Anomaly - Phantom Reads

```
/* Transaction 1 */
SELECT * FROM users WHERE age BETWEEN 10 AND 30;

                    /* Transaction 2 */
                    INSERT INTO users(id,name,age) VALUES ( 3, 'Bob', 27 );
                    COMMIT;

/* Transaction 1 */
SELECT * FROM users WHERE age BETWEEN 10 AND 30;
COMMIT;
```

References

https://en.wikipedia.org/wiki/Isolation_%28database_systems%29

http://www.firstsql.com/tutor5.htm

https://db.apache.org/derby/docs/10.3/devguide/cdevconcepts15366.html