CODING SHUTTLE

Transactions

Concurrent Transaction Management

Transaction Isolation

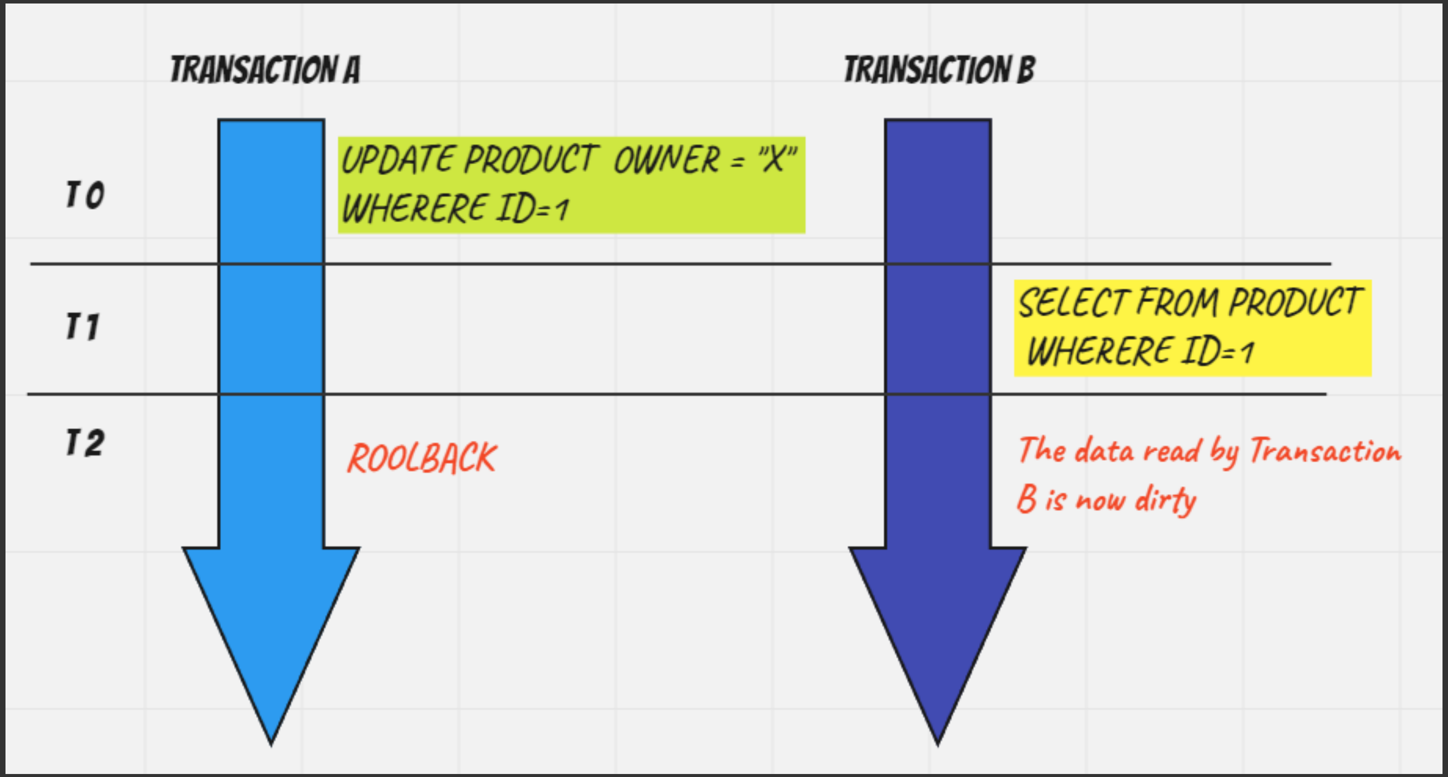
Isolation levels define the degree to which a transaction must be isolated from the data modifications made by any other transaction in the database system.

Spring Boot provides support for different isolation levels through the @Transactional annotation.

In order to understand isolation levels, we first need to look at some of the problems that occur in the database. Because isolation levels are a technology developed against these problems that we will talk about below.

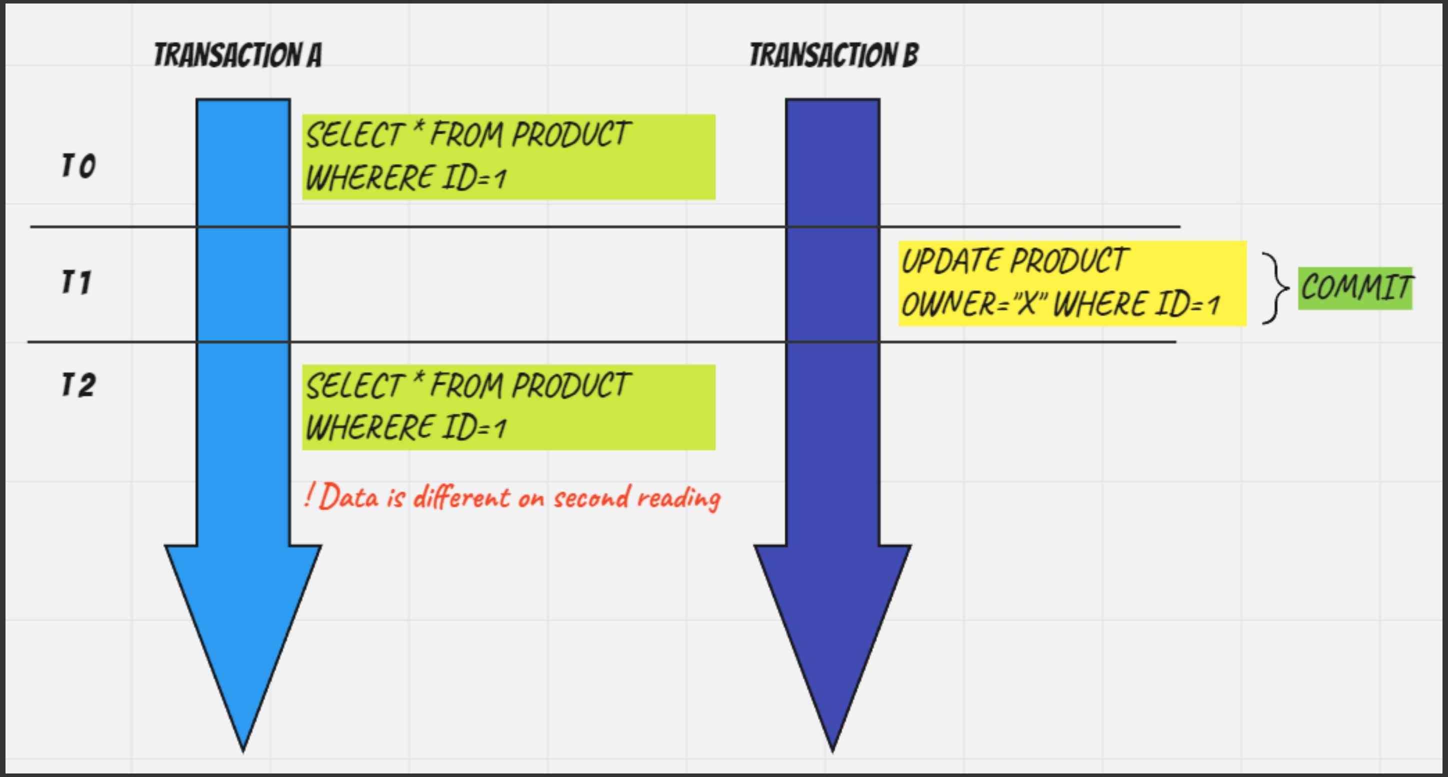
Dirty Read

Dirty Read - A Dirty read is a situation when a transaction reads data that has not yet been committed.



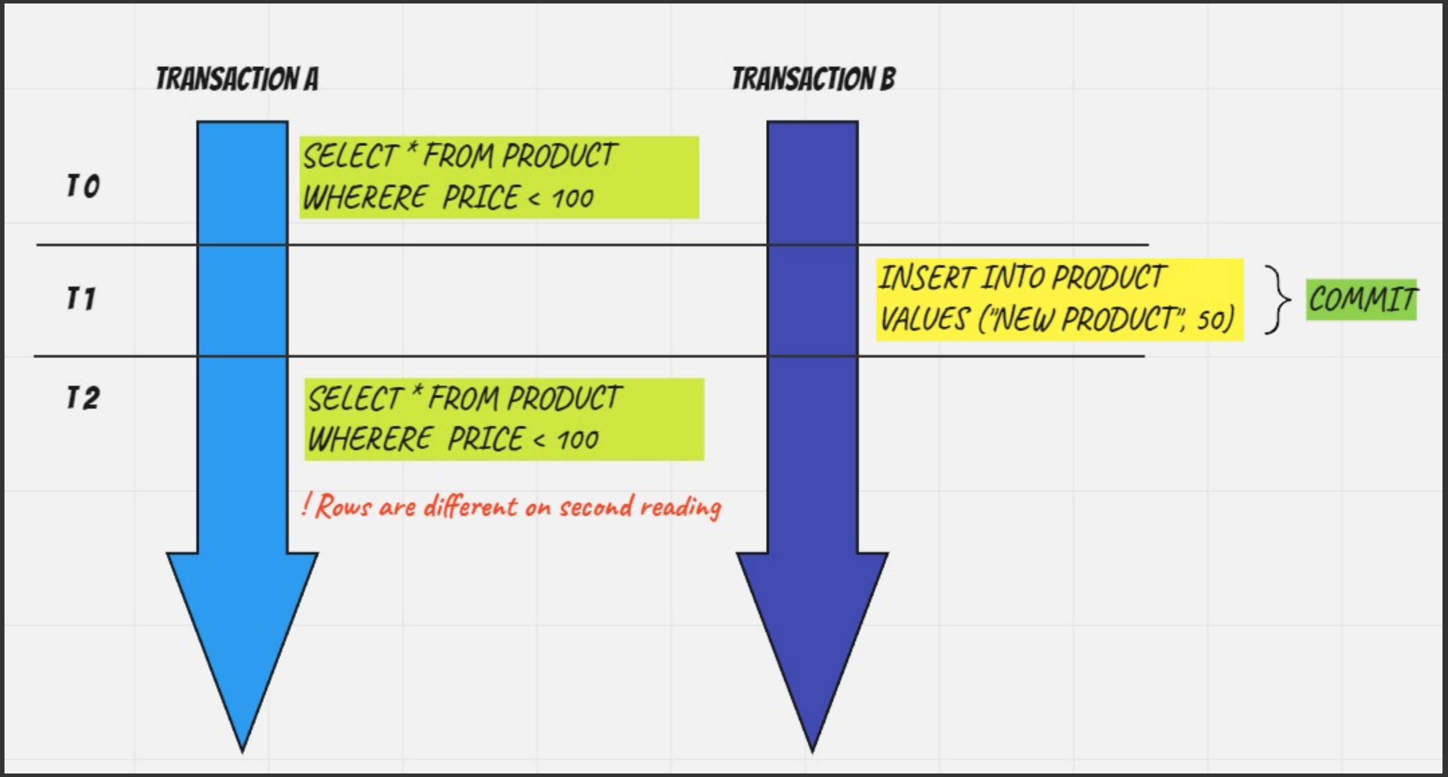
Non Repeatable read

Non Repeatable read - Non Repeatable read occurs when a transaction reads the same row twice and gets a different value each time.


TRANSACTION A
TRANSACTION B
T0
SELECT * FROM PRODUCT WHERERE ID=1
T1
UPDATE PRODUCT OWNER="X" WHERE ID=1
COMMIT
T2
SELECT * FROM PRODUCT WHERERE ID=1
! Data is different on second reading

Phantom Read

Phantom Read occurs when two same queries are executed, but the rows retrieved by the two, are different.



```
SELECT * FROM PRODUCT
WHERERE  PRICE < 100
```

```
INSERT INTO PRODUCT
VALUES ("NEW PRODUCT", 50)
```

```
SELECT * FROM PRODUCT
WHERERE  PRICE < 100
```

! Rows are different on second reading

Isolation Levels

- Read uncommitted
- Read committed
- Repeatable read
- Serializable

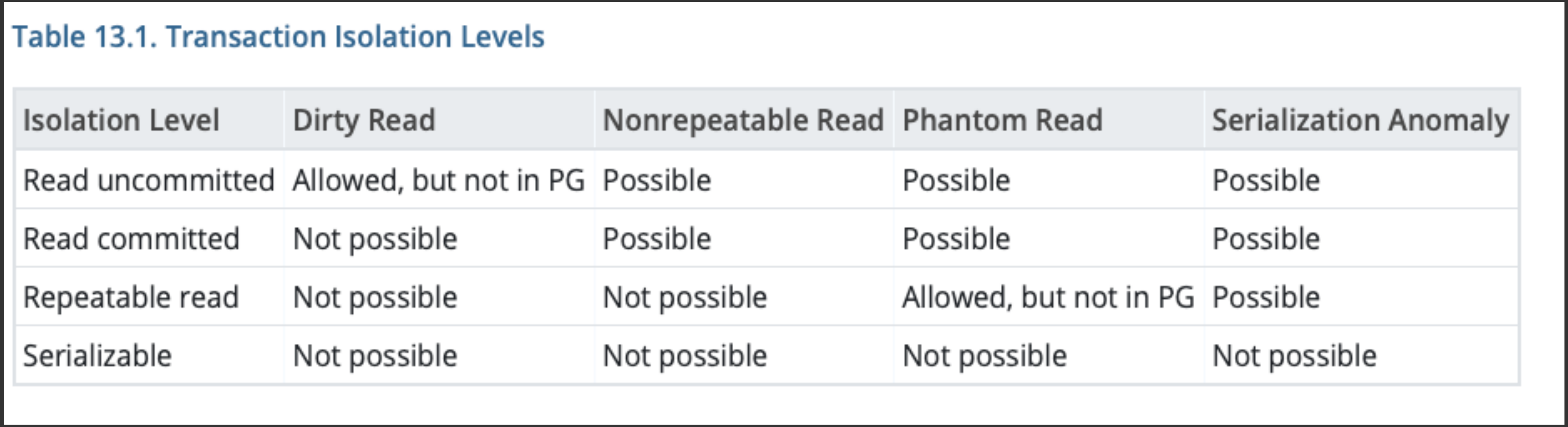


Table 13.1. Transaction Isolation Levels

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

How Serializable Isolation works

PostgreSQL uses a modern approach called Serializable Snapshot Isolation (SSI), which doesn't rely on strict locking but detects and aborts transactions that would lead to non-serializable behavior.

Oracle, MySQL, SQL Server, and DB2 rely on row and range locks, which ensure serializable behavior but can lead to decreased concurrency as transactions may need to wait for locks.