CS544 EA

Applications

Concurrency: Isolation Level

Isolation Levels

- Proper isolation is expensive (takes lots of time) to produce in a multi-user environment
  - Isolation is often relaxed to increase DB speed
  - ANSI SQL defines 4 isolation levels

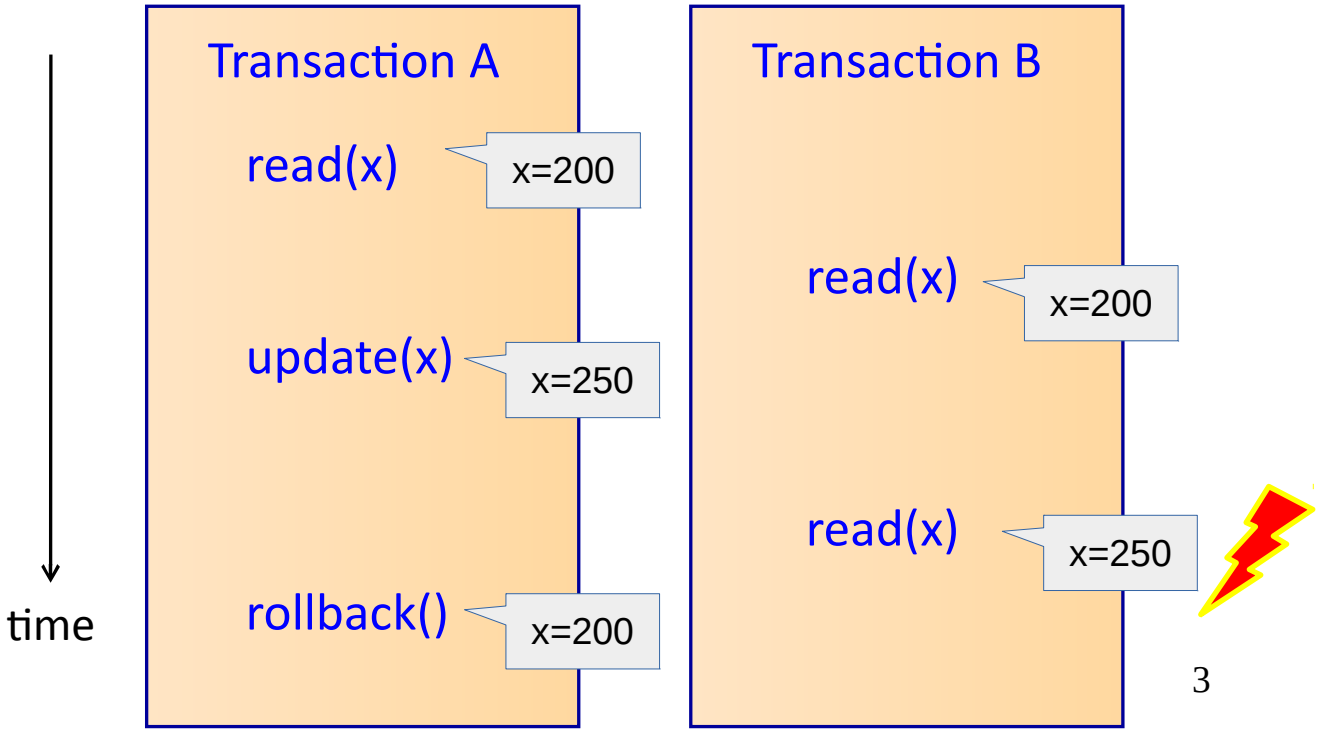
Read Uncommitted, Read Committed, Repeatable Read, Serializable

Weaker and Faster to Stronger and Slower

- Most Dbs default to Read Committed isolation
  - Only Serializable fully isolates a transaction from all concurrency issues

Read Uncommitted

- TX A can read TX B's uncommitted updates
  - No locks at all
  - **Violates ACID**
  - Not in Oracle
  - Don't use in concurrent env!

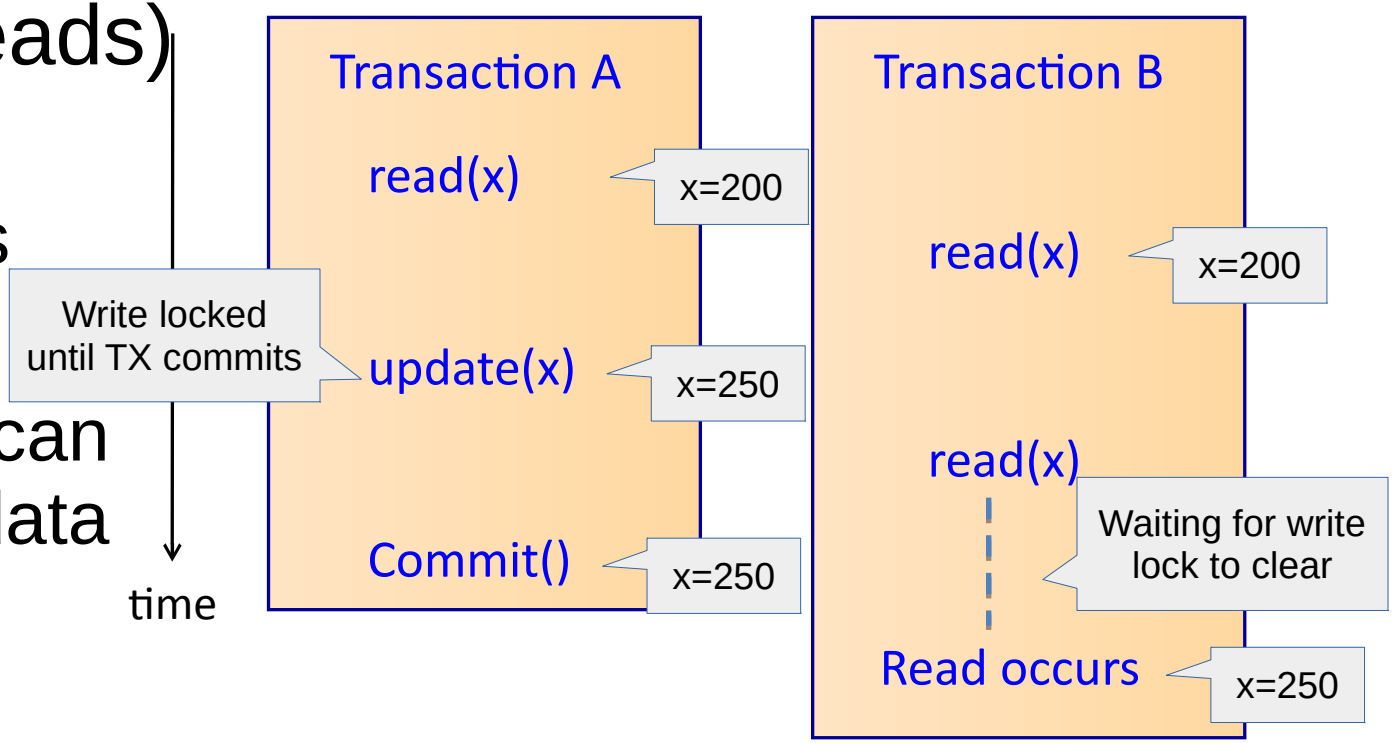


Concurrency Issues

- **Dirty reads**: a TX can read data that may never even get committed (useless)
- **Non-repeatable read**: a TX can read the same row twice and get two different values
- **Lost updates**: an update made by one TX silently disappears / overwritten (more on this later)
- **Phantom Read**: executing the same select twice may return more or less rows the second time

Read Committed

- Uses **write-locks** to hide non-committed data (solves dirty reads)
  - Written data blocks access until commit
  - Every thread can read normal data



Concurrency Issues

Not MySQL
it defaults to
Repeatable Read

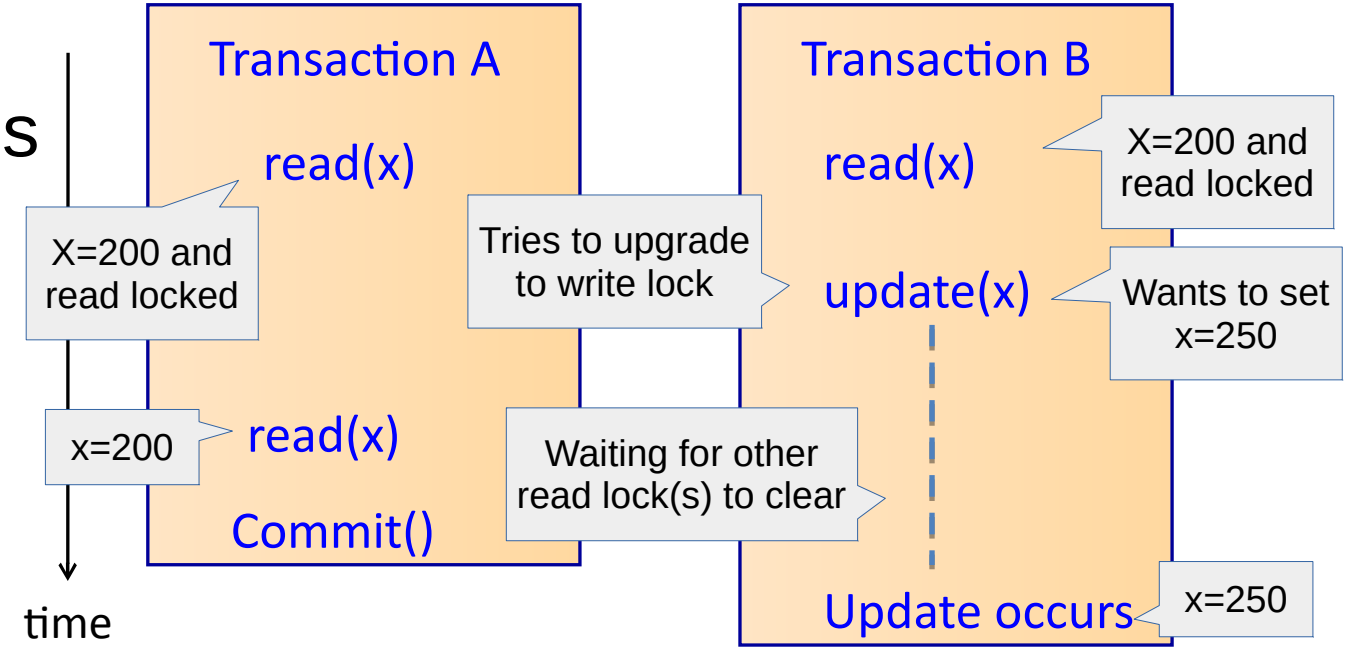
- Read Committed is the **Default for many Dbs**
  - Write locks cause some delays, but not significant
  - **Speed more important** than fixing concurrency issues

- Do provide are other ways of solving them:
  - Pessimistic locking (provided by most Dbs)
  - Optimistic concurrency (provided by JPA)

Not MySQL

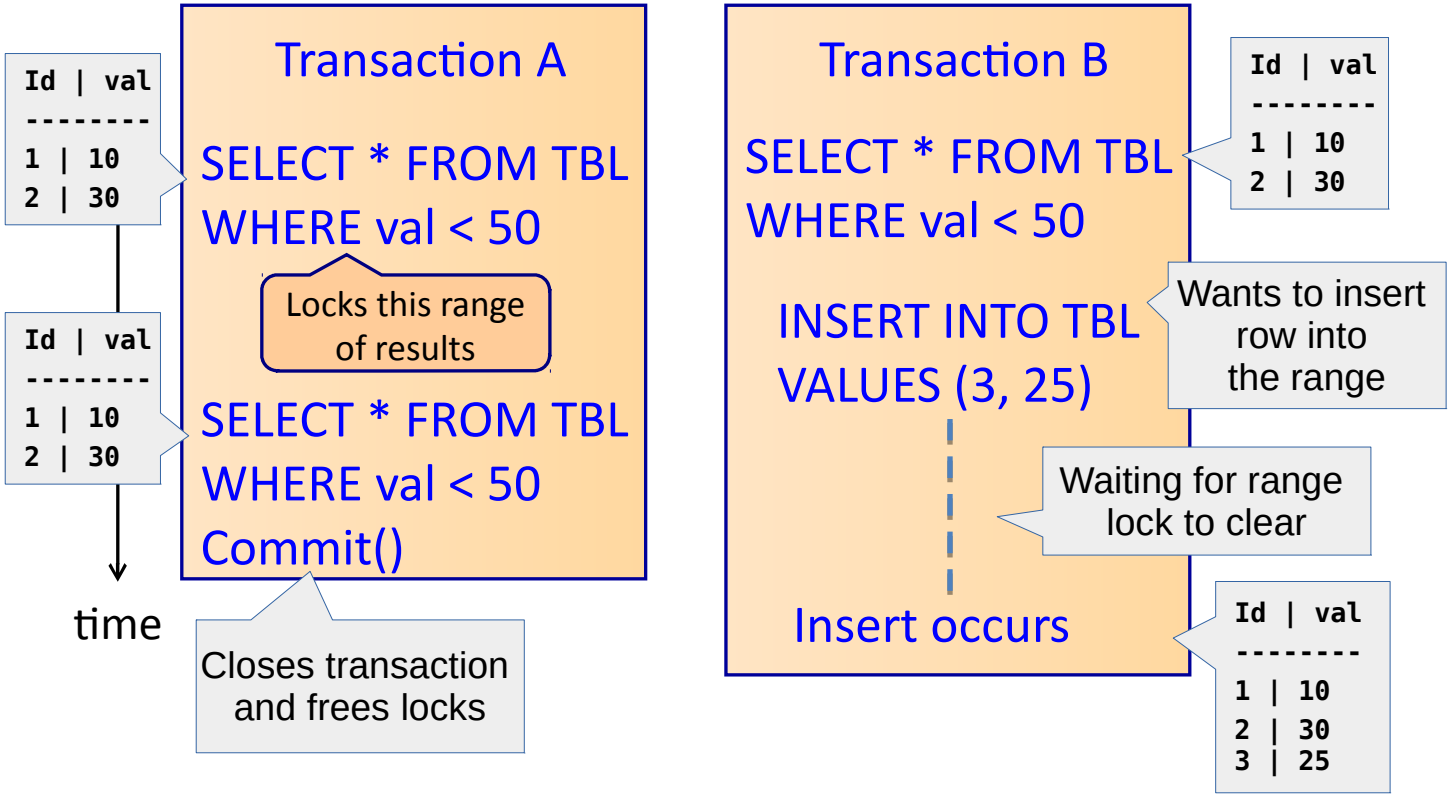
Repeatable Read

- **Uses read and write locks** to solve non-repeatable read and lost update problems
  - Once read all future reads same value



Serializable

- Sets **range locks** to solve phantom read
  - Lots of locks
  - Slow
  - Functionally similar to executing one after the other



Changing the default

- You can raise the default isolation level
  - **Everything will be slower**, less scalable
  - Even for transactions that don't need it

Not Recommended

- Inside persistence.xml:

```
<property name="hibernate.connection.isolation" value="8" />
```

1 – Read Uncommitted
2 – Read Committed
4 – Repeatable Read
8 – Serializable

Using read-committed

- Because **speed is usually more important** most databases use read-committed
- This leaves DBs open to:
  - Non-repeatable reads
  - Phantom reads
  - Lost update

Biggest Problem!

- We'll look at some ways to mitigate lost-update