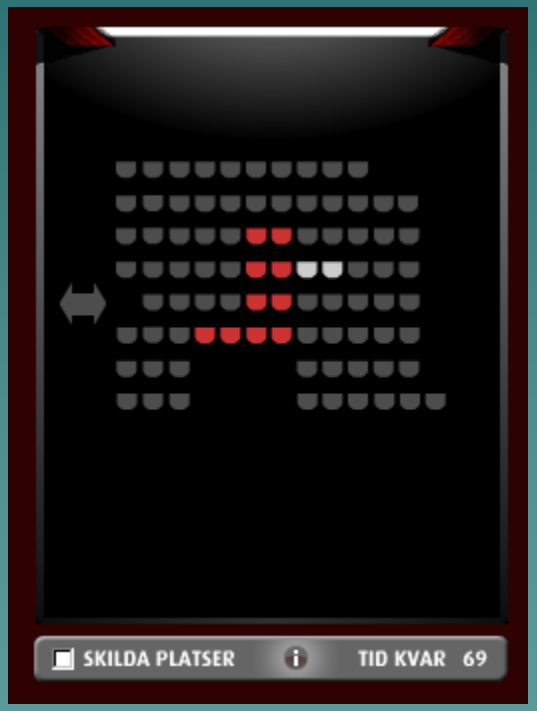
Exercise 6

Transactions
Isolation

Concurrency means trouble

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



Transactions

- START TRANSACTION
- COMMIT/ROLLBACK
- **Atomic**: Either the whole transaction is run, or nothing.
- **Consistent**: Database constraints are preserved.
- **Isolated**: Different transactions may not interact with each other.
- **Durable**: Effects of a transaction are not lost in case of a system crash.

Serializability

- If the outcome of a number of transactions is equal to the outcome of the transactions executed without time overlap

T_1	T_2	A	B
		25	25
READ(A,t)			
t := t+100			
WRITE(A,t)		125	
	READ(A,s)		
	s := s*2		
	WRITE(A,s)	250	
	READ(B,s)		
	s := s*2		
	WRITE(B,s)		50
READ(B,t)			
t := t+100			
WRITE(B,t)			150

To run transactions serially

- …would solve the problem…
- …but is often practically impossible
- Therefore we use different isolation levels

Locks

- To achieve different levels of isolation, you can use **locks**
- Shared (read) lock
- Exclusive (write) lock
- Locks are put on relevant parts of the data
- Lock requests are queued

Lock compatibility

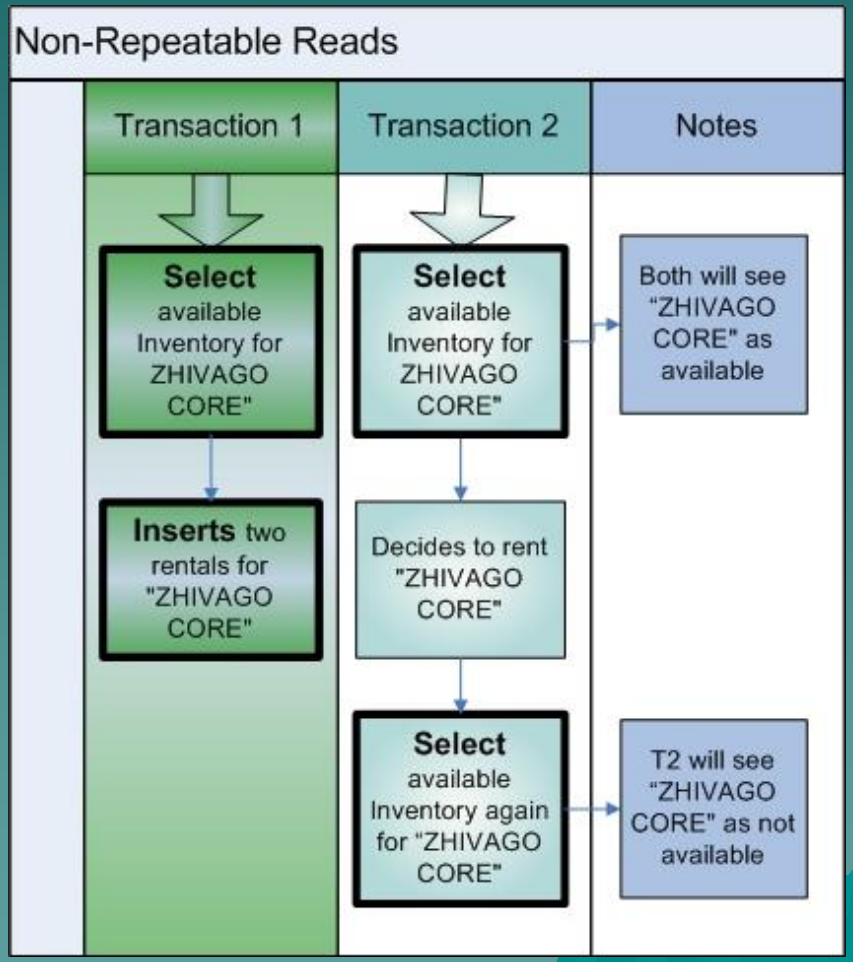
LOCK	Write	Read
Write	No	No
Read	No	Yes

Isolation 0 – read uncommitted

- **Does not ask** for read lock
- Therefore it can read data that another transaction has a **write lock** on
- ”**Dirty reads**” – data modified by another transaction, but not yet committed

Isolation 1 – read committed

- Asks for read lock, but **releases it after reading**
- **”Non-repeatable reads”** – the same query can give different results



Isolation 2 – repeatable read

- If a query has a **WHERE clause spanning a range**, a read lock is acquired only for the result, not the entire range (no **range lock**)
- The result can't be changed, but **new data can be added** – so called "**Phantoms**"

T1	T2
SELECT * **FROM** users **WHERE** age **BETWEEN** 10 **AND** 30;	
	INSERT INTO users **VALUES** (3, 'Bob', 27); COMMIT;
SELECT * **FROM** users **WHERE** age **BETWEEN** 10 **AND** 30;	

Isolation 3 - serializable

- No other transactions are allowed to interact with the data
- Range locks are used
- All locks collected are kept until after COMMIT

None of this is of course true…

- …at least not in ORACLE

Different approaches

”Pessimistic”

- ”We’d better make sure nothing funny is going on”
- Locks

”Optimistic”

- ”If something funny is going on, we can take care of that later”
- Check before COMMIT that everything is in order, otherwise abort

Which is better?

- Optimistic approach never blocks concurrent transactions
- But if conflicts happen often, the cost for aborting will be high

How to do it? (1st example) Timestamp ordering

- Every transaction is given a timestamp T_t
- Every object has two timestamps
  - Last read T_r
  - Last write T_w
- Read is only allowed if $T_t > T_w$, otherwise abort
  - Set $T_r = T_t$
- Write is only allowed if $T_t > T_r$, otherwise abort
  - Set $T_w = T_t$
  - SKIP write if $T_t < T_w$! (Thomas write rule)

How to do it? (2nd example) Multiversion concurrency control

- Several timestamped versions of data exist
- If $T_t > T_w$ the transaction can pick an older version to read, and does not have to abort (p 940)
- ORACLE uses something called Snapshot Isolation, based on MVCC
- A simple way to think of Oracle read consistency is to imagine each user operating a private copy of the database

Not even the locks are that simple…

	NL	SCH-S	SCH-M	S	U	X	IS	IU	IX	SIU	SIX	UIX	BU	RS-S	RS-U	RI-N	RI-S	RI-U	RI-X	RX-S	RX-U	RX-X
NL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
SCH-S	N	N	C	N	N	N	N	N	N	N	N	N	N	I	I	I	I	I	I	I	I	I
SCH-M	N	C	C	C	C	C	C	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
S	N	N	C	N	N	C	N	N	C	N	C	C	C	N	N	N	N	N	C	N	N	C
U	N	N	C	N	C	C	N	C	C	C	C	C	C	N	C	N	N	C	C	N	C	C
X	N	N	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C
IS	N	N	C	N	N	C	N	N	N	N	N	N	C	I	I	I	I	I	I	I	I	I
IU	N	N	C	N	C	C	N	N	N	N	N	C	C	I	I	I	I	I	I	I	I	I
IX	N	N	C	C	C	C	N	N	N	C	C	C	C	I	I	I	I	I	I	I	I	I
SIU	N	N	C	N	C	C	N	N	C	N	C	C	C	I	I	I	I	I	I	I	I	I
SIX	N	N	C	C	C	C	N	N	C	C	C	C	C	I	I	I	I	I	I	I	I	I
UIX	N	N	C	C	C	C	N	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
BU	N	N	C	C	C	C	C	C	C	C	C	C	N	I	I	I	I	I	I	I	I	I
RS-S	N	I	I	N	N	C	I	I	I	I	I	I	I	N	N	C	C	C	C	C	C	C
RS-U	N	I	I	N	C	C	I	I	I	I	I	I	I	N	C	C	C	C	C	C	C	C
RI-N	N	I	I	N	N	N	I	I	I	I	I	I	I	C	C	N	N	N	N	C	C	C
RI-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	N	N	N	C	C	C	C
RI-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	N	N	C	C	C	C	C
RI-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	N	C	C	C	C	C	C
RX-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C

Key

N	No Conflict	SIU	Share with Intent Update
I	Illegal	SIX	Shared with Intent Exclusive
C	Conflict	UIX	Update with Intent Exclusive
		BU	Bulk Update
NL	No Lock	RS-S	Shared Range-Shared
SCH-S	Schema Stability Locks	RS-U	Shared Range-Update
SCH-M	Schema Modification Locks	RI-N	Insert Range-Null
S	Shared	RI-S	Insert Range-Shared
U	Update	RI-U	Insert Range-Update
X	Exclusive	RI-X	Insert Range-Exclusive
IS	Intent Shared	RX-S	Exclusive Range-Shared
IU	Intent Update	RX-U	Exclusive Range-Update
IX	Intent Exclusive	RX-X	Exclusive Range-Exclusive