Relaxed Isolation, Isolation levels

- ANSI Isolation Levels
- Phenomena
- Preventing lost updates

transaction tuning, relaxed isolation

- Scheduling ensures serializability, but reduces *throughput,* the number of transactions per time unit (as measured for example in the TPC-C benchmark).
- Methods of increasing transaction thoughput:
- reduce isolation (reduce locking):
  - Reduced transaction isolation levels
  - Tables with lower isolation
  - Transaction chopping
  - Optimistic locking

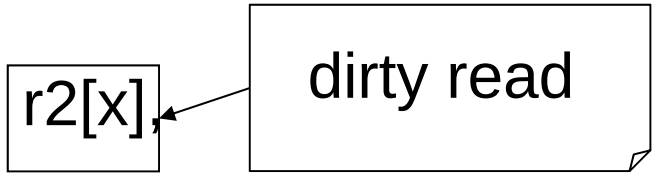
ANSI/ISO transaction isolation levels

- Isolation levels are defined with respect to three different phenomena (results of reduced isolation):
  - Dirty read: reading an uncommitted value for x.
  - Fuzzy read: reading different, committed values for x
  - Phantom: reading a new committed inserted row.
- ANSI/ISO SQL-92 defines four isolation levels:

Isolation Level	Dirty read	Fuzzy read	Phantom
READ UNCOMMITTED	Possible	Possible	Possible
READ COMMITTED	**Not** Possible	Possible	Possible
REPEATABLE READ	**Not** Possible	**Not** Possible	Possible
SERIALIZABLE	**Not** Possible	**Not** Possible	**Not** Possible

Dirty read (recap)

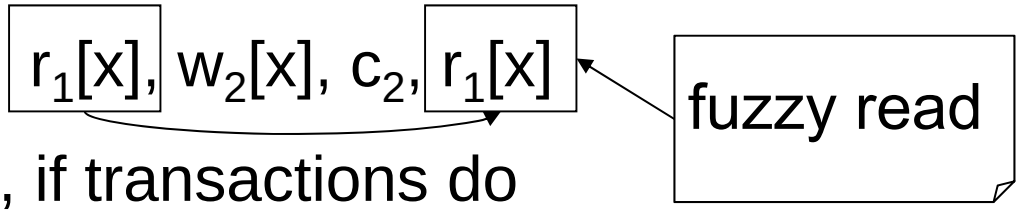
- Transaction TA2 performs a dirty read if it reads an uncommitted write result of TA1
  - scheduling example: ..... w1[x], r2[x],



- Dirty reads can happen, if transaction TA2 does not react to a write-lock on x.
- Dirty reads might be no problem for:
  - transactions that gather overview data
  - transactions that investigate options for later transactions
- But they are dangerous for other transactions
  - might lead to inconsistent results.
- Isolation level READ UNCOMMITTED allows dirty reads, but the transactions have to be read-only.

Fuzzy read

- Transaction TA1 encounters a fuzzy read phenomenon if it reads two or more different committed values for x.
  - scheduling example: $r_1[x], w_2[x], c_2, r_1[x]$



- Fuzzy reads can happen, if transactions do
  - not observe the read-lock of transaction TA1.
- Note, that fuzzy reads require two read operations by TA1, while dirty reads can happen with a single read.
- Fuzzy read situations can lead to lost updates in a rather counterfactual way:
  - If the second read does not happen or is not acted upon: (next slide)

Fuzzy read continued: lost updates

- a serious consequence of not using REPEATABLE READ:
  a committed transaction might miss an update: **lost update**

  $r_1[x], r_2[x], w_2[x], c_2, w_1[x], c_1$

  r1[x],

  here is the fuzzy read "zone"

  - example:
    - a123 is $99.
    - TA_1 withdraws $17, TA_2 withdraws $23
  - $r_1[x]$: $d_1 := 99$
  - $r_2[x]$: $d_2 := 99$
  - $w_2[x]$: a123 := 76 (== d_2-23) ← lost update
  - $w_1[x]$: a123 := 82 (== d_1-17)

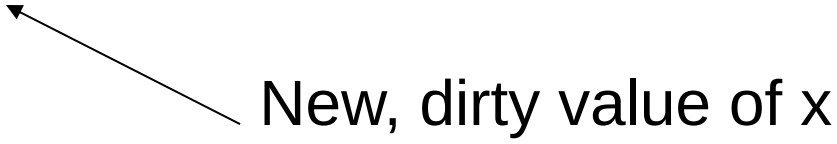
phantom

- A phantom (row) is a phenomenon that is possible in the relational data model, but goes beyond the basic read/write model.
- Are caused by inserts, not by updates.
- The following situation describes a phantom row:
  - TA1 performs: SELECT * FROM mytabl
    - gets a result set res1.
  - TA2 : inserts a row r into mytabl and commits.
  - TA1 performs again SELECT * FROM mytabl
    - gets a different result set res2 = res1 $\cup$ {r}
  - the row r is the phantom for TA1

Dirty read is worse than fuzzy read

- a case, where a transaction reads two different values, but one of them is a dirty value:

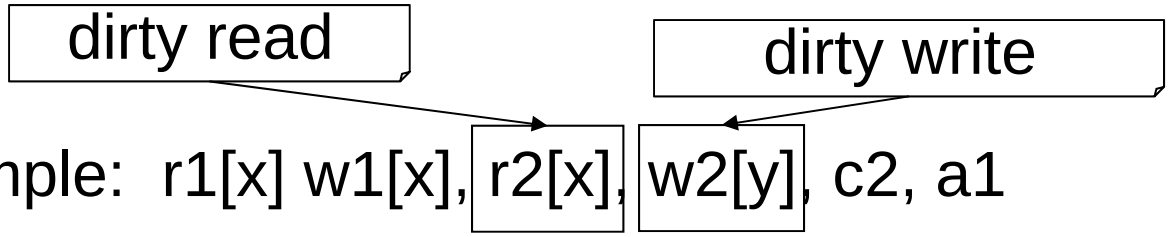
$r_1[x], r_2[x], w_2[x], r_1[x], w_1[x], c_1$



- This is called a dirty read and not a fuzzy read.
- Rationale: dirty read is a more serious phenomenon than fuzzy read.

dirty write following a dirty read

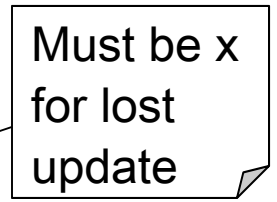
- Transactions in isolation level READ UNCOMMITTED can perform dirty reads, but are not allowed to write.
- Some DB's support an isolation level NONE, we assume this allows transactions to even write. Every write in a transaction following a dirty read we want to call a **dirty write**, since it can be influenced by the dirty read.
  - scheduling example: r1[x] w1[x], r2[x], w2[y], c2, a1



- Transaction level NONE is similar (but not identical) to a situation where all transactions have Autocommit=TRUE.
- A dirty write is a phenomenon that is more serious than a dirty read.

Describing phenomena:

- The names of the phenomena intuitively refer to one operation in the schedule, bolded in the following examples
- Fuzzy read: $r_1[x], r_2[x], w_1[x], c_1, \mathbf{r_2[x]}, w_2[z], c_2,$
- Lost update: $r_1[x], r_2[x], \mathbf{w_1[x]}, c_1 \quad w_2[x], c_2,$
- Dirty read: $r_1[x], r_2[x], w_1[x], \mathbf{r_2[x]}, r_2[z], c_2, c_1$
- Dirty write: $r_1[x], r_2[x], w_1[x], r_2[x], \mathbf{w_2[z]}, c_2, c_1$
- However, these operations are very variable for seemingly similar situations and therefore not good for identifying.
- Instead we talk about the **conflict object** (this is always **x** in the examples above), and the **conflict writing transaction** (here TA1) of the phenomenon.



Lost updates cannot be avoided by more reads

- The operations in the withdraw example:
  1. read a123 into local variable d,
  2. Check whether d greater than amount to be withdrawn
  3. IF yes: write b-amount to a123.
  4. commit
- Two alternatives in the basic transaction model:

	1.	2.	3.	4.
Alternative A: TA_1:	$r_1[x]$,		$w_1[x]$,	c_1
Alternative B: TA_1:	$r_1[x]$,		$r_1[x]$,$w_1[x]$,	c_1

- First question: what to do if the second read is different? Assume rollback.
- Can Alternative B prevent a lost update? No:

Lost updates

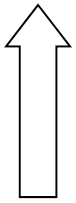
One case: second read of TA2 prevents lost update:

- s: $r_1[x], r_2[x], r_1[x], w_1[x], c_1, r_2[x], a_2$
- TA1: $r_1[x], r_1[x], w_1[x], c_1$
- TA2: $r_2[x], r_2[x]$ _____, a_2

(Rollback of TA2 because of changed value.)

Second case: Second read of TA2 happens slightly earlier: again a lost update happens.

- s: $r_1[x], r_2[x], r_1[x], r_2[x], w_1[x], c_1, w_2[x], c_2$
- TA1: $r_1[x], r_1[x], w_1[x], c_1$
- TA2: $r_2[x], r_2[x], w_2[x]$ _____, c_2

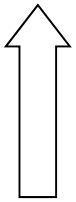


Preventing lost update in READ COMMITTED

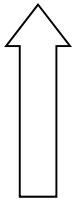
Explicitly getting an update lock with SELECT FOR UPDATE can prevent lost updates even in READ COMMITTED level.

Current situation:

- s: $r_1[x], r_2[x], w_1[x], c_1, w_2[x], c_2$
- TA1: $r_1[x], w_1[x], c_1$
- TA2: $r_2[x], w_2[x]$ _____, c_2



- Second read is now SELECT FOR UPDATE
- s: $R_1[x], w_1[x], c_1, R_2[x], w_2[x], c_2$
- TA1: $R_1[x], w_1[x], c_1$
- TA2: $R_2[x]$ _____, $w_2[x], c_2$



Summary

- We have seen the following bad phenomena of reduced isolation: phantom, fuzzy read, lost update, dirty read, dirty write.
- They are increasingly serious, except for the fuzzy read/lost update pair.
- They correspond to five increasingly relaxed isolation levels: SERIALIZABLE, REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, NONE.
- The first level SERIALIZABLE allows no phenomenon, the last level NONE allows all phenomena to occur.