

Exercise walk-through

8.5

Transactions and concurrency

University Computer Science

Transactions and concurrency: all or nothing

- 1 **Lost** T1 reads 100, T2 reads 100, T1 writes 90, T2 writes 120 - the withdrawal vanished: same race as the OS unit, now with money.
- 2 **Orders** R1 R2 W2 W1 ends 90 (deposit lost); R1 W1 R2 W2 ends 110 - equal to serial T1 then T2.
- 3 **Locks Two-phase locking** holds locks to commit: one waits, the outcome is serial-equivalent; deadlock is the price, retry the cure.

Interleaving timeline

same operations, two interleavings, two verdicts

lost update

T1 R(x) 100

T2 R(x) 100

T1 W(x) 90

T2 W(x) 120

ends 120: the -10 lost

serial-equivalent

T1 R(x) 100

T1 W(x) 90

T2 R(x) 90

T2 W(x) 110

ends 110: serial

2PL: reads and writes lock, locks held to commit - one transaction waits, the outcome equals a serial order

isolation levels name the anomalies you accept: dirty read gone at read-committed, phantoms gone only at serializable

ACID: atomic - consistent - isolated - durable

Two interleavings: one loses the update, one equals a serial order.

Exercises 2.1-2.3

2.1 ACID: name the C and what it means.

[3]

2.2 From 100, R2 R1 W1 W2 (withdraw 10, deposit 20): final value?

[3]

2.3 Name 2PL's price and its cure.

[3]

Exercises 2.1-2.3

2.1 ACID: name the C and what it means. [3]

2.1

- Consistency: constraints hold before and after every transaction ✓.

2.2 From 100, R2 R1 W1 W2 (withdraw 10, deposit 20): final value? [3]

2.2

- Both read 100; T2's write lands last: $100 + 20 = 120$ – the withdrawal lost ✓.

2.3 Name 2PL's price and its cure. [3]

2.3

- Deadlock; the detector kills a victim, which retries ✓.

Exercise 3.1

3.1 From balance 200: T1 withdraws 30, T2 deposits 50.

Compute the ending of R1 R2 W2 W1 and say which update vanished. Compute the ending of R1 W1 R2 W2. State the serializability verdict of each, with the reason. State what 2PL changes about the first schedule's outcome. [8]

Exercise 3.1

3.1 From balance 200: T1 withdraws 30, T2 deposits 50.

Compute the ending of R1 R2 W2 W1 and say which update vanished. Compute the ending of R1 W1 R2 W2. State the serializability verdict of each, with the reason. State what 2PL changes about the first schedule's outcome. [8]

3.1

- R1 R2 W2 W1: both read 200; final = $200 - 30 = 170$ – T2's deposit vanished ✓.
- R1 W1 R2 W2: $170 + 50 = 220$ ✓.
- Verdicts: the first is not serializable (no serial order yields 170); the second equals serial T1-then-T2 ✓.
- 2PL: T2's read of the locked balance waits, so the schedule becomes the serial-equivalent one ending 220 ✓.

Exercise 3.2

3.2 T1 updates rows matching a filter; T2 inserts a new matching row between T1's two reads.

Name the anomaly. State which isolation level prevents it and which lowest level already shows consistent re-reads. Give the mechanism serializable uses (range locks or equivalent) and why read-committed cannot stop the insert. [7]

Exercise 3.2

3.2 T1 updates rows matching a filter; T2 inserts a new matching row between T1's two reads.

Name the anomaly. State which isolation level prevents it and which lowest level already shows consistent re-reads. Give the mechanism serializable uses (range locks or equivalent) and why read-committed cannot stop the insert. [7]

3.2

- The phantom read: T1's second read sees a row its first did not ✓.
- Serializable prevents it; repeatable-read already gives consistent re-reads of existing rows ✓.
- Serializable uses range locks over the predicate's key range (or an equivalent) – the insert's key falls inside a held range ✓.
- Read-committed locks only current row reads – no range is held, so the matching insert proceeds ✓.