Ibrahim Johar Farooqi (23K-0074)
BAI-5A
Database Systems — Assignment 04

day / date:

Q.1)

a) The ACID Properties violated here are Atomicity and Consistency. As only partial part of the transaction took place, and it did not complete, breaking atomicity. Consistency is violated as well, since the database is now left in an inconsistent state.

b) The violation happened as the transaction that included the 'driver accept + wallet deduction + confirmation' did not complete a proper atomic commit before the crash. The driver acceptance and wallet deduction were written to the database but before the final ride confirmation, the server crashed leaving the transaction partially executed. This broke Atomicity because the operation did not complete as 'all or nothing' factor/ruling, and it broke consistency as money was deducted even though there was no confirmation of the ride.

c) A customer books a ride, which starts a single database transaction. The system reserves the payment from the wallet, records the driver's acceptance, and prepares the ride confirmation. All these operations are performed within one atomic transaction. If every step completes successfully, the system will issue a COMMIT & the confirmation will be shown to customer. If for any reason the server crashes, before the commit happens, then the transaction is rolled back during recovery using Log Records. As a result, the wallet balance would be restored, the driver's acceptance would rolled back (undo) and no confirmation would be sent to the customer. This approach prevents partial updates & ensures that all ACID properties are maintained.

KAGHAZ
www.kaghaz.pk

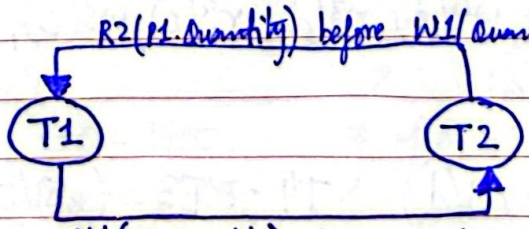
Scanned with CamScanner

d) The long term business consequences of this would be (if this continues to happen), customers will lose trust in the business due to wrong charges, or missing confirmations. These chargebacks and refunds would increase costs, as well as, an increase in operational overheads. Legal exposure, and negative business image could become a central issue for the business.

Q.2) a) non-serial schedule: R1(P1.Quantity); R2(P1.Quantity); W1(P1.Quantity − Sold_units); W2(P1.Quantity + return_units);

b) operations in order:
- ↳ R1(Q) [T1]
- ↳ R2(Q) [T2]
- ↳ W1(Q) [T1]
- ↳ W2(Q) [T2]

conflicting pairs (edge)
- ↳ R1(Q) before W2(Q) [T1 → T2]
- ↳ R2(Q) before W1(Q) [T2 → T1]
- ↳ W1(Q) before W2(Q) [T1 → T2]




W1(P1.Quantity) before W2(P1.Quantity) [write-write]
R1(P1.Quantity) before W2(P1.Quantity) [read-write]

c) The schedule is not conflict serializable, as the precedance graph contains a cycle (T1 → T2), indicating that there is no equivalent serial order without conflicts.

d) serializing T1 before T2
Corrected Schedule: R1(P1.Quantity); W1(P1.Quantity − sold_units); R2(P1.Quantity); W2(P1.Quantity + return_units);

e) Real world loss from this anomaly could look like loosing customer trust, damage to reputation, as the business misrepresents inventory numbers, like displaying overstocking or understocking of products. Ultimately could result in loss in sales, as customers and clients would be disappointed.

KAGHAZ www.kaghaz.com
Scanned with CamScanner

| T1 | T2 | T3 | T4 |
|---|---|---|---|
| r1(a) | r2(b) | r3(c) | r4(d) |
| r1(b) | r2(c) | r3(d) | r4(a) |
| w1(a) | w2(b) | w3(c) | w4(d) |

day / date:

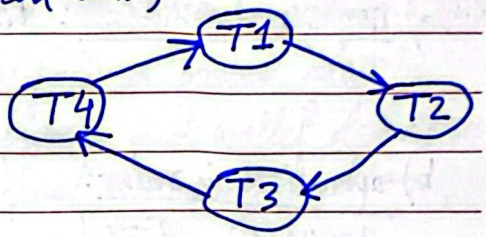
Q.3)

a) S1 : r1(a), r2(b), r3(c), r4(d), r1(b), r2(c), r3(d), r4(a), w2(b), w3(c),
1) w4(d), w1(a)

| | | |
|---|---|---|
| r4(a) → w1(a) | T4 → T1 | (read-write) |
| r3(d) → w4(d) | T3 → T4 | (read-write) |
| r2(c) → w3(c) | T2 → T3 | (read-write) |
| r1(b) → w2(b) | T1 → T2 | (read-write) |

T4 → T1 → T2 → T3 → T4

↳ cycle exists in S1.



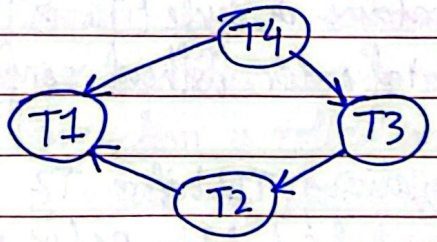

2) ↳ S1 is not conflict serializable

3) ↳ None exist

S2 : r4(d), r4(a), w4(d), r3(d), r3(c), w3(c), r2(b), r2(c), w2(b), r1(a),
1) r1(b), w1(a).

| | | |
|---|---|---|
| w4(d) → r3(d) | T4 → T3 | (write-read) |
| r4(a) → w1(a) | T4 → T1 | (read-write) |
| w2(b) → r1(b) | T2 → T1 | (write-read) |
| w3(c) → r2(c) | T3 → T2 | (write-read) |

T4 → T3 → T2 → T1
→ T1




2) ↳ Yes, S2 is conflict Serializable, as it is Acyclic.

3) ↳ equivalent serial schedule:
↳ T4 → T3 → T2 → T1

KAGHAZ
www.kaghaz.pk

Scanned with CamScanner

T1: 2000 − 300 = 1700
T2: 2000 + 800 = 2800

day / date:

Q.4)

1) The schedule faces the Lost Update problem, as both transactions read the same initial value (2000), before any transaction writes back its update. So when T1 writes 1700, later T2 writes 2800, based on the same initial value, this causes the transactions to overwrite each transaction's update. This is a result of the absence of any synchronization or locking mechanism to prevent the transactions from altering the same data simultaneously.

2) final amount after both transactions finish: 2800
↳ as T2 overwrote T1's 1700.

3) if transactions were correctly executed, serially T1 → T2 and T2 → T1, then.

↳ case: T1 → T2
* initial value: 2000
* after T1: 2000 − 300 = 1700
* after T2: 1700 + 800 = 2500
↳ final result: 2500

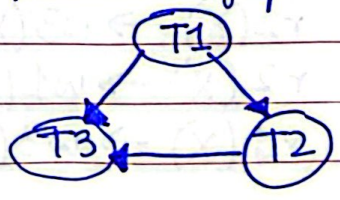
↳ case: T2 → T1
* initial val: 2000
* after T2: 2000 + 800 = 2800
* after T1: 2800 − 300 = 2500
↳ final result: 2500

Q.5) a) r1(X); w1(X); r3(X); r2(X); w3(X);

↳ conflicts & edges

| | |
|---|---|
| w1(X) → r3(X) | T1 → T3 ✓ |
| w1(X) → r2(X) | T1 → T2 ✓ |
| w1(X) → w3(X) | T1 → T3 |
| r2(X) → w3(X) | T2 → T3 ✓ |
| r1(X) → w3(X) | T1 → T3 |

precedance graph




no cycle.

↳ Conflict Serializable (Acyclic)
↳ Serial Schedule: T1 → T2 → T3

KAGHAZ
www.kaghaz.pk
Scanned with CamScanner

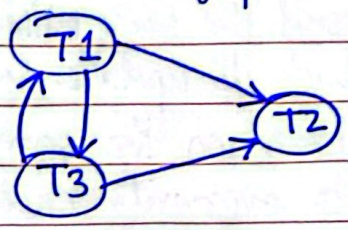
day / date:

b) r1(X); r3(X); w3(X); w1(X); r2(X);

↳ edges & conflicts

| | |
|---|---|
| r1(X) → w3(X) | T1 → T3 |
| r3(X) → w1(X) | T3 → T1 |
| w3(X) → r2(X) | T3 → T2 |
| w1(X) → r2(X) | T1 → T2 |

precedance graph




cycle exist

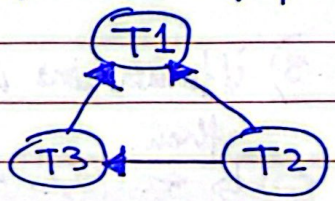
↳ Not conflict serializable (Has cycle)

c) r3(X); r2(X); w3(X); r1(X); w1(X);

↳ conflicts & edges

| | |
|---|---|
| r3(X) → w1(X) | T3 → T1 |
| r2(X) → w3(X) | T2 → T3 |
| w3(X) → r1(X) | T3 → T1 (same) |
| r2(X) → w1(X) | T2 → T1 |

precedance graph




no cycle (Acyclic)

↳ Conflict Serializable

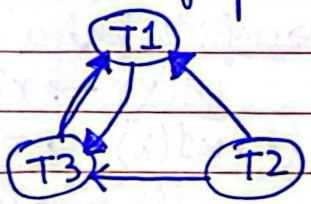
↳ equivalent serial order: T2 → T3 → T1

d) r3(X); r2(X); r1(X); w3(X); w1(X);

↳ conflicts & edges

| | |
|---|---|
| r3(X) → w1(X) | T3 → T1 |
| r2(X) → w3(X) | T2 → T3 |
| r2(X) → w1(X) | T2 → T1 |
| r1(X) → w3(X) | T1 → T3 |

precedance graph




cycle exists

↳ Not Conflict Serializable (Has Cycle)

KAGHAZ
www.kaghaz.pk

Scanned with CamScanner