23K-0801
BAI-5A

DBMS - Assignment - 4

Q1)

a) (1) Atomicity: The transaction did not complete fully; one part (wallet deduction) succeeded while the other (ride confirmation) did not.

(2) Consistency: The system ended up in an invalid state money deducted but no confirmed ride.

(3) Durability: The system failed to ensure that the correct final result (ride creation + wallet deduction) was stored persistently before the crash.

b) The failure occured because the step involved in booking the ride were not executed as single atomic transaction. Since these were executed in separate database calls or microservice transactions, the server crashed between steps.

c) (1) The customer requests a ride.

(2) The system opens a single atomic transaction.
Step A: Reserve (not deduct) wallet balance in a pending state.
Step B: Create a ride record with status "Pending Confirmation".

(3) Only if both step A & step B succeed, the transaction is commited.

(4) After commit:
- The app shows the ride confirmation to the customer.
- The reserved amount is converted to a final deduction when the ride starts.

2315-0001

d) (1) Financial losses: Users charged without receiving a service cause forced refunds, disputes, & support overhead.
(2) Loss of custom trust, users uninstall the app if money disappears without confirmed rides.
(3) ~~Fees~~ drivers think they accepted rides that customers never saw. Leads to wasted time & frustration.

Q2)

a) non-serial schedule:
r1 (P1.quantity), r2 (P1.quantity), w1 (P1.quantity-solid-units)
w2(P1.quantity + returned_units)

b)
$$\begin{tikzcd} T1 \arrow[r, "\text{w1 (P1.quantity) before w2 (P1.quantity) (write-write)}"] & T2 \arrow[l, bend left, "\text{r2 (P1.quantity) before w1 (P1.quantity) (read-write)}"] \end{tikzcd}$$

c) The schedule is not conflict serializable because the precedence graph contains a cycle, indicating no equivalent serial order without conflicts.

d) Corrected schedule:
r1 (P1.quantity), w1 (P1.quantity-solid-units), r2 (P1.quantity),
w2 (P1.quantity + returned_units)

c) Inventory accuracies could lead to overstocking / understocking, causing lost sales or excess holding costs. This might result in financial losses from return / refunds, damaged customer relationships.

Q3)

a)

$$\begin{tikzcd} T4 \arrow[r] & T1 \arrow[dl] \\ T2 \arrow[r] & T3 \arrow[ul] \end{tikzcd}$$

(b)

$$\begin{tikzcd} T1 & T2 \arrow[l] \\ T3 \arrow[ur] & T4 \arrow[l] \end{tikzcd}$$

b) Not conflict-serializable (cycle) | conflict serializable (acyclic)

c) Not serializable | The serial schedule is T4→T3→T2→T1

Q4)

a) The concurrency problems are lost update & dirty read (as more precisely a write-write conflict leading to lost update). They arise because both transactions read the same initial value (2000) without isolation, compute independently & overwrite each other.

b) The final amount is 2800 after both transactions finish.

c) T1 → T2

2000 - 300 => 1700 + 800 => 2500

T2 → T1

2000 + 800 => 2800 - 300 => 2500

231K-0001

ہجری ۱۴۳___ Lunar — —143

عیسوی ۲۰۱___ Solar — —201

Q5)

a)

$$\begin{tikzcd} T1 \arrow[rr] \arrow[dr] & & T2 \arrow[dl] \\ & T3 & \end{tikzcd}$$

Conflict serializable (acyclic)

Serial schedule : T1 → T2 → T3

(b)

$$\begin{tikzcd} T1 \arrow[r] \arrow[d, bend left] & T2 \\ T3 \arrow[u, bend left] & \end{tikzcd}$$

Not conflict serializable (cyclic)

Not conflict serializable (cycle)

c)

$$\begin{tikzcd} T1 & & T2 \arrow[ll] \arrow[dl] \\ & T3 \arrow[ul] & \end{tikzcd}$$

Conflict Serializable (acyclic)

Serial schedule : T2 → T3 → T1

(d)

$$\begin{tikzcd} T1 \arrow[dr, bend right] & & T2 \arrow[ll] \arrow[dl] \\ & T3 \arrow[ul, bend right] & \end{tikzcd}$$

Not conflict serializable (cyclic)