# DB Assignment

23100069

Q1 a) atomicity was violated : some sub-operations were applied while the overall logical transaction did not complete ( no confirmed booking shown)

Durability may also be violated if the server crash occured before changes were flushed to stable storage committed, state might be lost

b) They system performed multiple related writes across components without a single atomic transaction spanning them

The application didn't use a transactional mechanism (e.g. database transaction with begin / commit) then a crash between partial writes leaves the system in consistent state

c) Server begin transaction
driver accepted ride Customer wallet balance was deducted server crashed DBMS sees transactions that ended without commit, it will roll back

d) loss of customer, financial loss, Damage to brand

Q2

(a) Let initial quantity be Q. Let sold units in $T_1$ be s and returned units in $T_2$ be r

- A common problematic interleaving is
  $r_1(Q), r_2(Q), w_1(Q := Q-s), w_2(Q := Q+r)$
  (read by $T_1$, read by $T_2$, then $T_1$ writes, then $T_2$ writes)

(b) . conflicts occur when two diff transactions access the same item and atleast one access is a write

Examining the interleaving:

1) $r_1(Q)$ then $r_2(Q)$ - read/read: no conflict
2) $r_1(Q)$ then $w_1(Q)$ - same transaction ignore
3) $r_1(Q)$ then $w_2(Q)$ - read by $T_1$ before write by $T_2$: conflict $\Rightarrow T_1 \rightarrow T_2$
4) $r_2(Q)$ then $w_1(Q)$ read by $T_2$ before write by $T_1$ conflict $T_2 \rightarrow T_1$
5) $w_1(Q)$ then $w_2(Q)$ - write/write conflict $\Rightarrow T_1 \rightarrow T_2$ already present

Graph  cycle

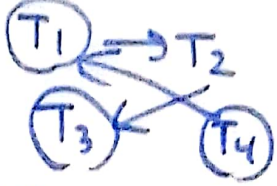


(c) has a cycle the schedule is not conflict serializable

(d) option 1 $T_1$ then $T_2$: → option 2 $T_2$ then $T_1$:
$r_1(Q), w_1(Q := Q-s), r_2(Q), w_2(Q := Q+r)$ → $r_2(Q), w_2(Q := Q+r), r_1(Q), w_1(Q := Q-s)$
final quantity becomes $Q-s+r$

e) Incorrect inventory Customer dissatisfaction / Revenue loss

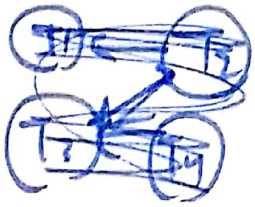
Q3




23IC0069

S1 : precedence Edges $T_1 \to T_2$ ⇒, $T_2 \to T_3$, $T_3 \to T_4$, $T_4 \to T$
→ cycle → not serializable

S2 : precedence edges $T_4 \to T_3$, $T_2 \to T_3$, $T_2 \to T_1$, $T_4 \to T_1$ →
acyclic → serializable

Example equivalent serial serial schedules
$T_2 \to T_4 \to T_3 \to T_1$ or $T_4 \to T_2 \to T_3 \to T_1$ (both valid as they respect the Edges

Q4

1) lost update (write-write) anamoly : both $T_1$ and $T_2$ reads the same initial value (2000) and computes updates independently. $T_1$ writes 1700 then $T_2$ overwrite that with 2800 - $T_1$ update is effectively lost
Non-serializable interleaving: The interleaving does not correspond to any serial Execution of $T_1$ and $T_2$. The lack of isolation leads to an inconsistent outcome

2) 2800 ($T_2$'s write overwrote $T_1$'s write)

3) If $T_1$ then $T_2$
After $T_1$: 2000 − 300 = 1700
then $T_2$ adds 800: 1700 + 800 = 2500
If $T_2$ then $T_1$
After = 2800
Then $T_1$ applies discount : 2800 − 300 = 2500
final 2500

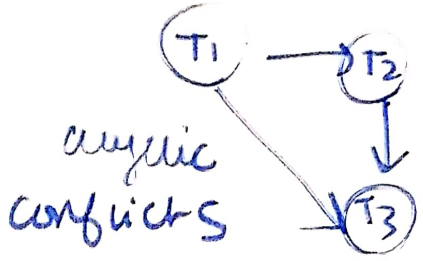
Q5

23/10/06

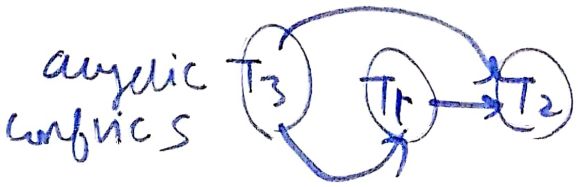
(a) Serializable Eq. S : $T_1 \to T_2 \to T_3$
(b) Serializable Eq. S : $T_3 \to T_1 \to T_2$
(c) Serializable Eq. S : $T_2 \to T_3 \to T_1$
(d) not conflict-serializable (cycle b/w $T_1$ and $T_3$)

(a) schedule $r_1(X)$ ; $w_1(X)$ ; $r_3(X)$ ; $r_2(X)$ $w_3(X)$

Conflict edges $T_1 \to T_3$, $T_1 \to T_2$, $T_2 \to T_3$

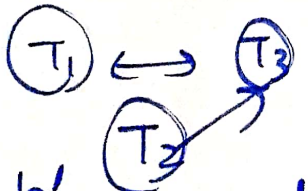



(b) Edges from conflict $T_3 \to T_1$ , $T_3 \to T_2$ , $T_1 \to T_2$




(c) Edges from conflict $T_2 \to T_3$, $T_3 \to T_1$


(d) edges from conflict $T_2 \to T_3$, $T_1 \to T_3$, $T_3 \to T_1$




Cycle b/w $T_1$ and $T_3$ hot conflict Serializable.