Naman Jain Vimal Kumar
1001670153

1) ***Discuss the atomicity, durability, isolation, and consistency preservation properties of a database transaction.***

Atomicity:
In a transaction involving multiple discrete pieces of information, either complete set of instructions are committed, or none are.

Durability:
A successful transaction creates a new and valid state of data, while a failed transaction returns all data to its state before the transaction was started.

Isolation:
Transactions that are on process and not committed yet remain isolated from other transactions.

Consistency:
Only committed data is stored in the system, such that even in the event of failure or system restart, the data is available in its current and valid state.

2) ***Discuss the different types of failures. What is meant by catastrophic failure?***

Computer Failure:
A failure in hardware, software or network of the computer system during transaction execution.

Transaction or System Error:
Some operations like integer overflow or division by zero causes transaction failures. Such transactions are termed as Transaction or System Error. They also include logical programming error or erroneous parameter values.

Logical Error or Exception Conditions detected by the transaction:
A transaction that proceeds but stops or cancels all inputted data due to error during execution. For example, amount withdrawal can cause problem if current amount is less. These errors can be handled in transaction to avoid transaction failure.

Concurrency Control Enforcement:
The concurrency control method may call off a transaction as it violates serializability. Or it may abort one or more transactions to resolve the deadlock state among several transactions.

Disk Failure:
It is not possible to read or write data if there is a read-write header malfunction or a crash with the same. This leads to loss of information.

Physical Problems and Calastrophes:
It includes forms of physical misfortune to our database server. There could be multiple problems, some of them includes power or air-conditioning failure, fire, theft, sabotage, overwriting of disks by mistake, and mounting of wrong tape by the operator.

3) Discuss the actions taken by the read_item and write_item operations on a database.

Note: Let the item to read be 'X'
Actions taken by the read_item operation in database:

1. Find the address of the disk block that contains the item X.
2. Copy the disk block into a buffer in the main memory, if not present in main memory buffer.
3. Copy item X from the buffer to the program variable.

Note: Let the write operation be performed on 'X'
Actions taken by the write_item operation in database:

1. Find the address of the disk block that contains the item X.
2. Copy the disk block into a buffer in the main memory, if not present in main memory buffer.
3. Copy item X from the program variable named X into its correct location in the buffer.
4. Store the updated block from the buffer back to disk.

4) What is a serial schedule? What is a serializable schedule? Why is a serial schedule considered correct?

Serial Schedule:
A schedule 'S' is termed Serial if, for every transaction 'T' participating in the schedule 'S', all the operations of 'T' are executed consecutively in the schedule 'S'.

Serializable Schedule:
Types of schedules that are always considered to be correct when concurrent transactions are executing are known as serializable schedules.

Serial Schedule is correct:
As we know, in a serial schedule only one complete transaction is processed then the others are followed. There isn't any interleaving occurrence. Every transaction is independent, as every transaction is correct when executed separately. Since every transaction is executed from beginning to end in complete isolation, we can say that serial schedule is correct.

5) Draw a state diagram and discuss the typical states that a transaction goes through during execution

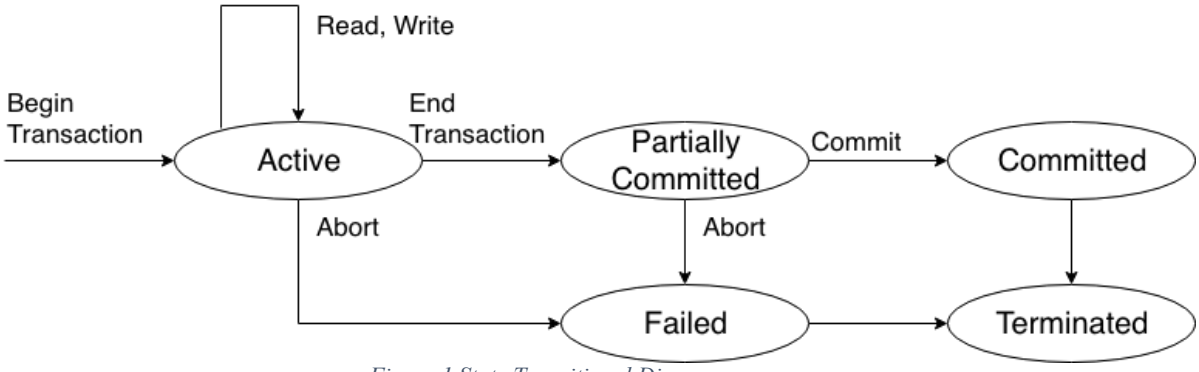

Figure 1 State Transitional Diagram

Active State:
A transaction enters the active state immediately after it starts execution. Here it can execute its READ and WRITE operations. It leaves the current state, once the transaction is ended.

Partially Committed State:
A transaction enters into the partially committed state once the transaction is ended by the active state. Here, some recovery protocols like recording in the system log are ensured such that a system failure will not result in an inability to record the changes of the transaction permanently. Once it is done, a transaction is committed and leaves the state.

Committed State:
A transaction enters into the committed state once the transaction is committed. Note, when a transaction is committed, it has concluded its execution successfully and all its changes must be recorded permanently in the database, even if a system failure occurs.

Failed State:
A transaction enters into the failed state if one of the checks fails or if the transaction is aborted during its active state. The transaction here are rolled back to undo the effects of WRITE operations on the database.

Terminated State:
This state corresponds to the transactions leaving the system. The entries maintained with respect to the transaction are removed from the transaction information table. Failed or aborted transactions may be restored later automatically or manually as brand new transactions.

6) Define the violations caused by each of the following: dirty read, non-repeatable read, and phantoms

Dirty Read:
The problem is occurred when a transaction ‘A’ updates the value ‘X’ and transaction ‘B’ reads the updated value of ‘X’, but then transaction ‘A’ fails. In the Figure 2, Transaction ‘B’ reads ‘X’ which was updated by transaction ‘A’, which failed. The value read by transaction ‘B’ is called as dirty read as it was created by a transaction that was not committed yet.

Non-Repeatable Read:
A transaction ‘A’ reads ‘X’, and transaction ‘B’ writes a new value to ‘X’. Now when transaction ‘A’ reads ‘X’ again, it sees up a different value.

Phantoms:
This is similar to a non-repeatable read, but here ‘X’ consists of the records that satisfy a WHERE condition. When a transaction ‘A’ reads ‘X’, and transaction ‘B’ writes a record satisfying the same WHERE conditions. But when transaction ‘A’ reads ‘X’ yet again, it sees a new record which is called the phantoms.

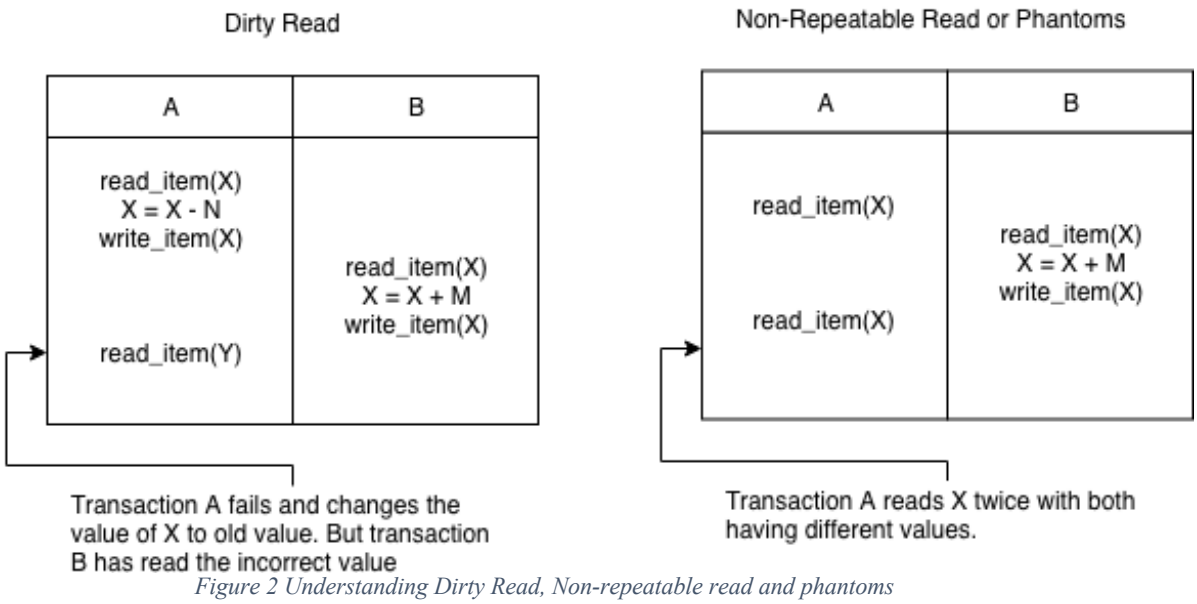

Figure 2 Understanding Dirty Read, Non-repeatable read and phantoms

7) Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below. Draw the Serializability (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s)

T1: r1(x); r1(z); w1(x)
T2: r2(z); r2(y); w2(z); w2(y)
T3: r3(x); r3(y); w3(y)

S1: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y)
S2: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y)

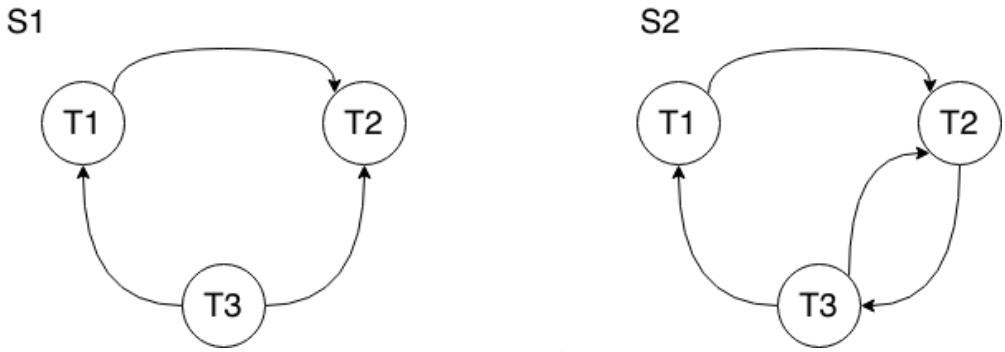

Figure 3 Precedence Graph for S1 and S2

S1 is serializable as it does not produce a cycle, thus we can say that it is conflict serializable. Equivalent serial schedule [T3, T1, T2] for the same is:

r3(x), r3(y), w3(y), r1(x), r1(z), w1(x), r2(z), r3(y), w3(y)

S2 produces a cycle, thus has a conflict with serialization.

8) Which of the following schedules is (conflict) serializable (use precedence graphs)?
a) r1(X) r3(X) w1(X) r2(X) w3(X)
b) r3(X) r2(X) w3(X) r1(X) w1(X)

c) r3 (X) r2 (X) r1(X) w3(X) w1(X)
d) r2(Z) r2(Y) w2(Y) r3(Y) r3(Z) r1(X) w1(X) w3(Y) w3(Z) r2(X) r1(Y) w1(Y) w2(X)
e) r3(Y) r3(Z) r1(X) w1(X) w3(Y) w3(Z) r2(Z) r1(Y) w1(Y) r2(Y) w2(Y) r2(X) w2(X)

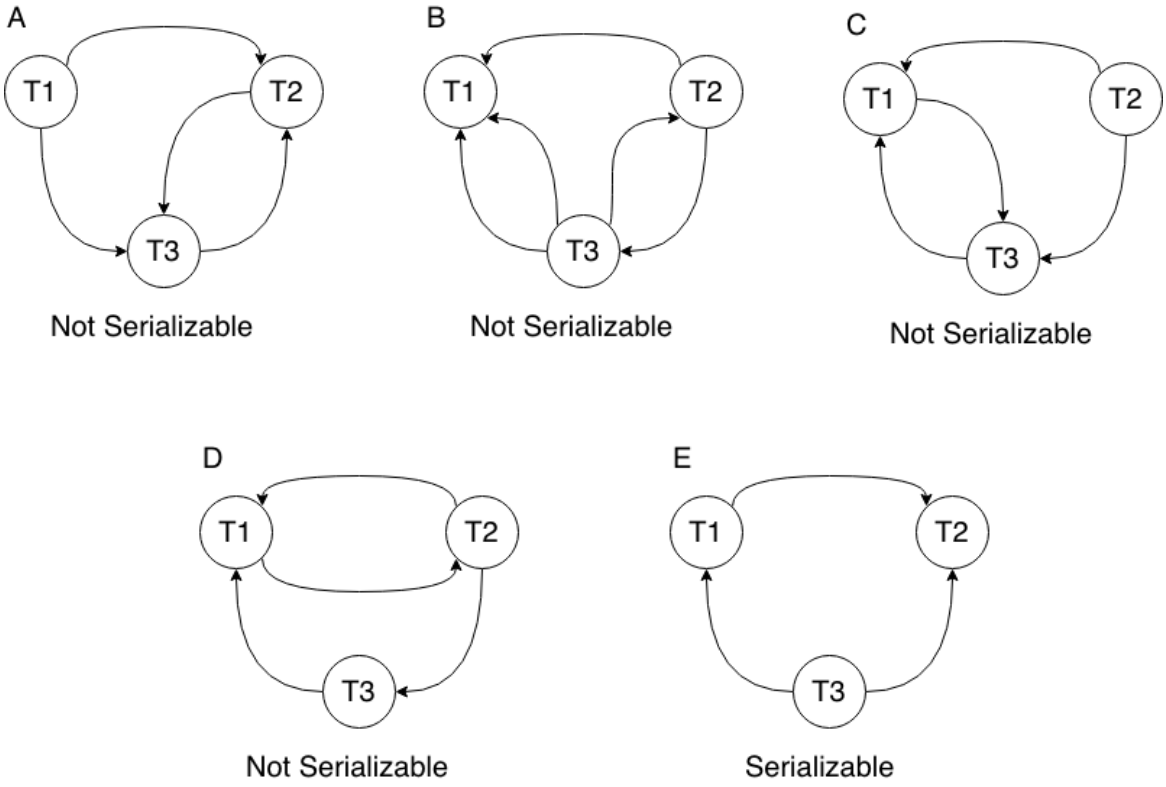

Figure 4 Answer to Question 8

9) Consider schedules S3, S4, and S5 below. Determine whether each schedule is strict, Cascadeless, recoverable, or non-recoverable. (Determine the strictest recoverability condition that each schedule satisfies.) Hint: c=commit.

S3: r1(X) r2(Z) r1(Z) r3(X) r3(Y) w1(X) c1 w3(Y) c3 r2(Y) w2(Z) w2(Y) c2
S4: r1(X) r2(Z) r1(Z) r3(X) r3(Y) w1(X) w3(Y) r2(Y) w2(Z) w2(Y) c1 c2 c3
S5: r1(X) r2(Z) r3(X) r1(Z) r2(Y) r3(Y) w1(X) c1 w2(Z) w3(Y) w2(Y) c3 c2

Properties	*S3*	*S4*	*S5*
Recoverable	True	False Due to dirty read W3(Y) and R2(Y)	True
Cascadeless	True	False Due to dirty read W3(Y) and R2(Y)	True
Strict	True	False Due to R2(Y) after W3(Y) without C3	False Due to W2(Z) after W3(Z) without C3

10) What is the total number of possible schedules for the three transactions in figure below? Please write the correct formula.

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

Number of operations in T1: 4
Number of operations in T2: 5
Number of operations in T3: 4
Total number of operations in a schedule = 13

We know that,

$$\#schedules = \frac{(\#operation_{total})!}{\prod_{i=transactions}(\#operation_i)!}$$

$$\#schedules = \frac{13!}{4! * 5! * 4!}$$

$$\#\boldsymbol{schedules = 90090}$$

References:

- *https://searchsqlserver.techtarget.com/definition/ACID*
- *https://www.youtube.com/watch?v=ivMPRcg90Xc*
- *Fundamentals of Database Systems, Sixth Edition by Elmasri/Navathe.*