Ch - 6

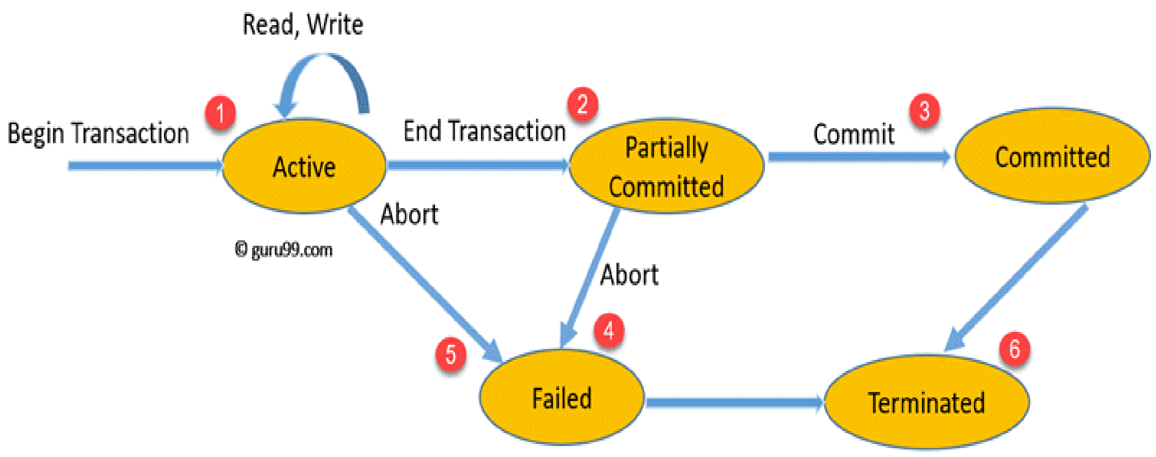
Transactions Management , Concurrency and Recovery

What is a Database Transaction?

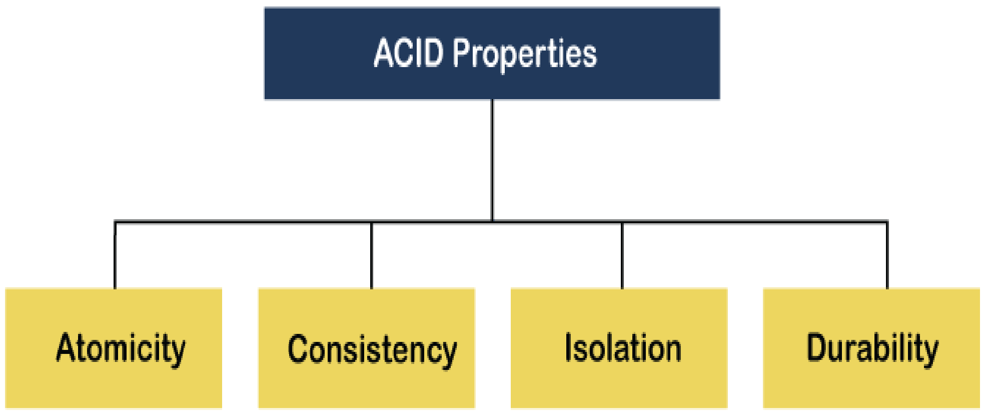
A Database Transaction is a logical unit of processing in a DBMS which entails one or more database access operation.

Why do you need concurrency in Transactions?

A database is a shared resource accessed. It is used by many users and processes concurrently. For example, the banking system, railway, and air reservations systems, stock market monitoring, supermarket inventory, and checkouts, etc.



ACID Properties



1) Atomicity:

The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all.

It means that the operation should not break in between or execute partially.

In the case of executing operations on the transaction, the operation should be completely executed and not partially.

2) Consistency:

The word consistency means that the value should remain preserved always.

In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always.

In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

3) Isolation:

The term 'isolation' means separation.

In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently.

The operation on one database should begin when the operation on the first database gets complete.

It means if two operations are being performed on two different databases, they may not affect the value of one another.

In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained.

Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

4) Durability:

Durability ensures the permanency of something.

In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database.

The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.

However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

2. Concurrency Control

It is the method of managing concurrent operations on the database without getting any obstruction with one another.

3. Transaction Control

The following commands are used to control transactions.

COMMIT − to save the changes.

ROLLBACK − to roll back the changes.

SAVEPOINT − creates points within the groups of transactions in which to ROLLBACK.

SET TRANSACTION − Places a name on a transaction.

What is Transaction?

A set of logically related operations is known as a transaction. The main operations of a transaction are:

Read(A): Read operations Read(A) or R(A) reads the value of A from the database and stores it in a buffer in the main memory.

Write (A): Write operation Write(A) or W(A) writes the value back to the database from the buffer.

Let us take a debit transaction from an account that consists of the following operations:

```
R(A);
A=A-1000;
W(A);
```

Assume A's value before starting the transaction is 5000.

The first operation reads the value of A from the database and stores it in a buffer.

The Second operation will decrease its value by 1000. So buffer will contain 4000.

The Third operation will write the value from the buffer to the database. So A's final value will be 4000.

Consider the following transaction involving two bank accounts x and y:

```
read(x);

x := x – 50;

write(x);
```

```
read(y);

y := y + 50;

write(y);
```

The constraint that the sum of the accounts x and y should remain constant is that of ?

Atomicity

Consistency

Isolation

Durability

Lock-based-protocols

There are two types of lock:

1. Binary Locks − A lock on a data item can be in two states; it is either locked or unlocked.

2. Shared lock:

- It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
- It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

3. Exclusive lock:

- In the exclusive lock, the data item can be both reads as well as written by the transaction.
- This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

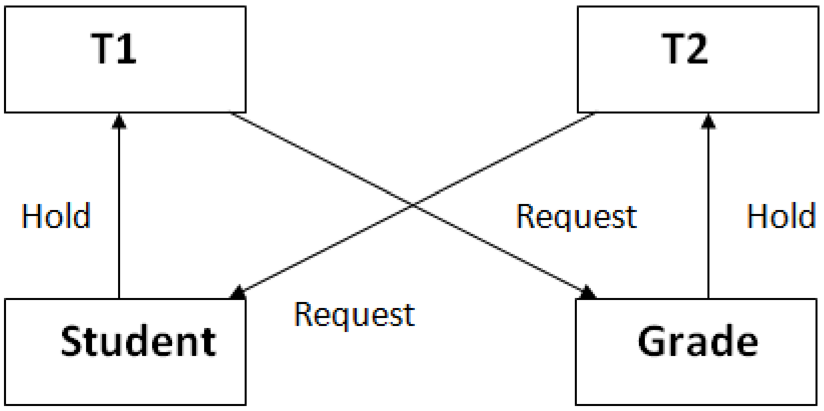
Types of lock protocols available

Deadlock handling

Deadlock is a state of a database system having two or more transactions, when each transaction is waiting for a data item that is being locked by some other transaction.

For example: In the student table, transaction T1 holds a lock on some rows and needs to update some rows in the grade table. Simultaneously, transaction T2 holds locks on some rows in the grade table and needs to update the rows in the Student table held by Transaction T1.

Now, the main problem arises. Now Transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 to release its lock. All activities come to a halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transactions.



Timestamp-based protocols

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.

- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.

- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.

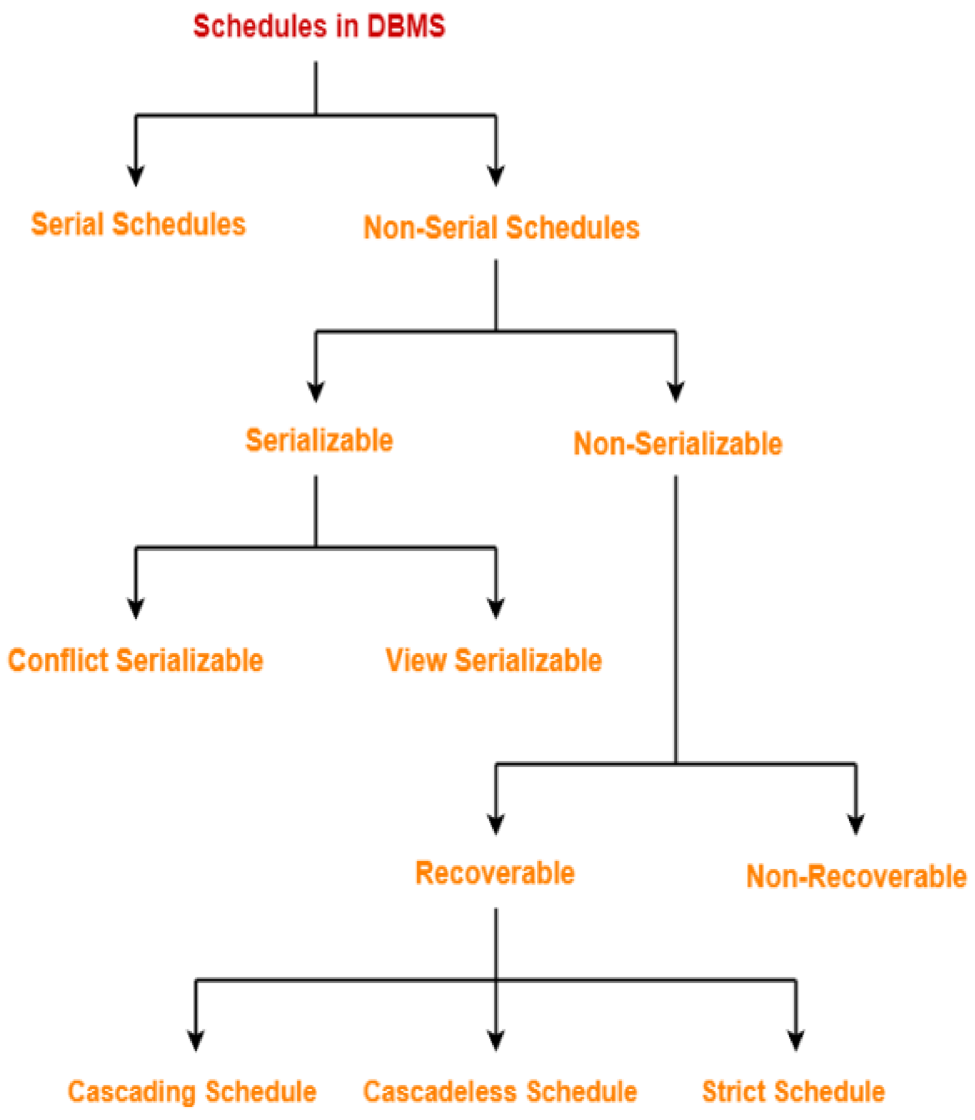
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at

009 times. T1 has the higher priority, so it executes first as it is entered the system first.

- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

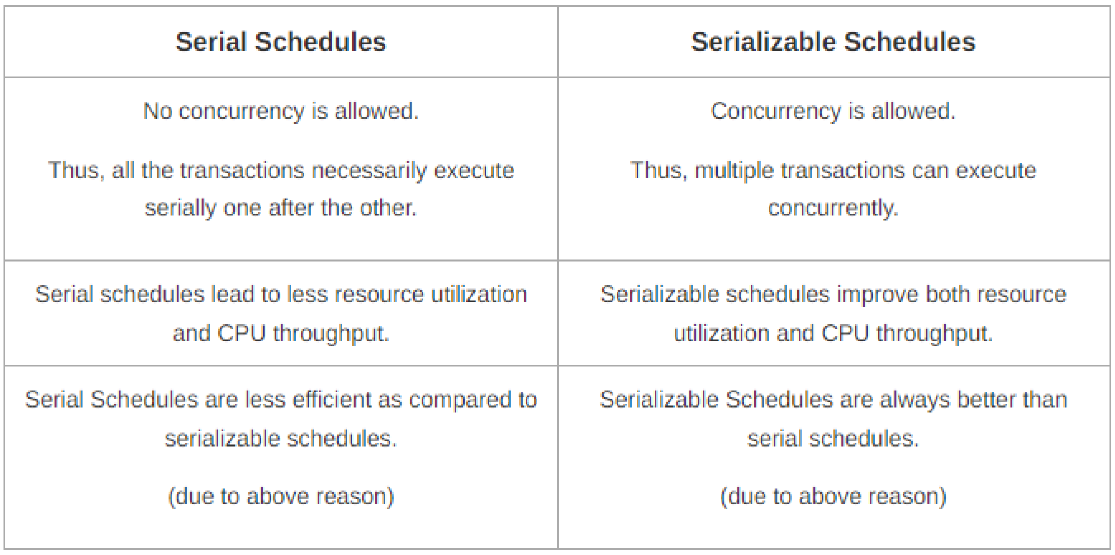
Schedule

- A schedule is the order in which the operations of multiple transactions appear for execution.
- Serial schedules are always consistent.
- Non-serial schedules are not always consistent.

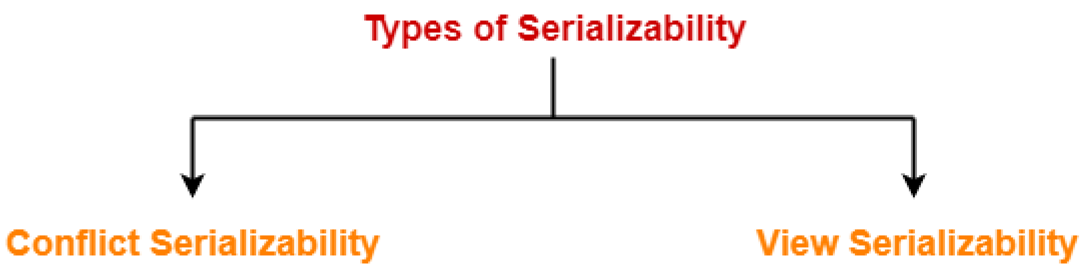


- Serializability is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database.

Serial Schedules Vs Serializable Schedules-



Serial Schedules	Serializable Schedules
No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other.	Concurrency is allowed. Thus, multiple transactions can execute concurrently.
Serial schedules lead to less resource utilization and CPU throughput.	Serializable schedules improve both resource utilization and CPU throughput.
Serial Schedules are less efficient as compared to serializable schedules. (due to above reason)	Serializable Schedules are always better than serial schedules. (due to above reason)



Conflict Serializability-

- If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called as a conflict serializable schedule.

Conflicting Operations-

- Two operations are called as conflicting operations if all the following conditions hold true for them-
    - Both the operations belong to different transactions
    - Both the operations are on the same data item
    - At least one of the two operations is a write operation

Example-

Consider the following schedule-

Transaction T1	Transaction T2
R1 (A)	
W1 (A)	
	R2 (A)
R1 (B)	

- In this schedule,

- R1 (B) and R2 (A) are called as conflicting operations.
- This is because all the above conditions hold true for them.

<u>View Serializability -</u>

- A schedule will view serializable if it is view equivalent to a serial schedule.
- If a schedule is conflict serializable, then it will be view serializable.
- The view serializable which does not conflict serializable contains blind writes.

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:

1. Initial Read

An initial read of both schedules must be the same. Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A)	
	Write(A)

Schedule S1

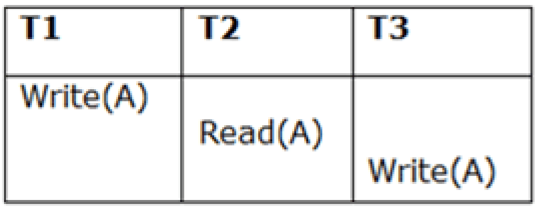
T1	T2
	Write(A)
Read(A)	

Schedule S2

Above two schedules are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

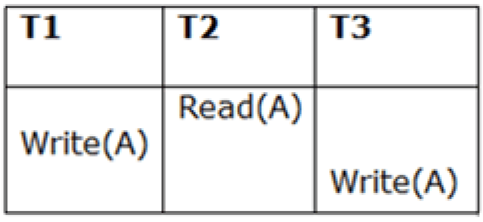
2. Final Write

A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.



T1	T2	T3
Write(A)		
	Read(A)	
		Write(A)

Schedule S1



T1	T2	T3
	Read(A)	
Write(A)		
		Write(A)

Schedule S2

Above two schedules is view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

Recovery System

What is recovery?

- Recovery is the process of restoring a database to the correct state in the event of a failure.
- It ensures that the database is reliable and remains in consistent state in case of a failure.

Database recovery can be classified into two parts;

1. Rolling Forward applies redo records to the corresponding data blocks.
2. Rolling Back applies rollback segments to the datafiles. It is stored in transaction tables.

Log-Based Recovery

- Logs are the sequence of records, that maintain the records of actions performed by a transaction.
- In Log – Based Recovery, log of each transaction is maintained in some stable storage. If any failure occurs, it can be recovered from there to recover the database.
- The log contains the information about the transaction being executed, values that have been modified and transaction state.
- All these information will be stored in the order of execution.

Example:

Assume, a transaction to modify the address of an employee. The following logs are written for this transaction,

Log 1: Transaction is initiated, writes 'START' log.
Log: <Tn START>

Log 2: Transaction modifies the address from 'Pune' to 'Mumbai'.
Log: <Tn Address, 'Pune', 'Mumbai'>

Log 3: Transaction is completed. The log indicates the end of the transaction.
Log: <Tn COMMIT>