Shortcuts

I/O = Input Output

Transaction Management

A transaction is defined as any one execution of a user program in a DBMS and differs from a execution of a program outside the DBMS

The Asset Property

A DBMS must ensure four important properties of transactions to maintain data in the phase of conquering access and system failures:-

1. Atomic
2. Consistency
3. Isolation
4. Durability

Atomic

Users should be able to regard the execution of each transaction as Atomic i.e either all actions are carried out or none are

Consistency

Each transaction run by itself with no conquering execution of other transactions must preserve the Consistency of the database

Isolation

Users should be able to understand a transaction without considering the effect of other conquer my executing transactions even if the DBMS interleaves the actions of several transactions for performance reasons this property is referred to as Isolation

Durability

Once the DBMS informs the user that a transaction has been successfully completed its effects should persist even if their system crashes before all its changes are reflected on disk. This is called Durability

Transactions and Schedules

- A transaction is seen by the DBMS as a series or list of actions the actions that can be executed by a transaction includes reads and writes of database

objects

- To keep our motions simple we assume that an object O is always read into a program variable that is also named O. We can therefore denote the action of transaction T reading an object O as RT(O). Similarly we can denote WT(O)
- In addition to reading and writing each transaction must specify as its final action either commit(Complete Successfully) or abort(Terminate and Undo all the actions carried out thus far)
- Abort T denotes the action of T aborting and commit T denotes T committing, we make two important assumptions

1. Transactions interact with each other via only database read and write operations. Ex:- They are not allowed to make messages
2. A database is a fixed independent objects when objects are added or deleted from a database or their are relationship with between database objects that we want to exploit for performance some additional issues arise that's when come Schedules come into actions

Schedules

A Schedule is a list of actions(Read, Write Abort or Commit) from a set of transactions and the order in which to actions of a transaction appear in a Schedule must be same as the order in which they appear in T we emphasize that a Schedule describes a transaction as seen by the DBMS

- In addition to these actions a transaction may carry out other action to other operating files evaluating mathematical expressions however these actions do not affect other transactions

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)
R(C)	
W(C)	

Conquering execution of transactions

Now we have the introduced the concept of a Schedule we have convenient way to describe interleave of a transaction

The actions of different transactions but not all interleave should be allowed

Motivation for conquer execution

The Schedule shown in the figure represents a interleaved execution of two transactions

1. While one transaction is waiting for one page to be from disk the cpu can process another transaction this is because I/O can be done in parallel with cpu activity on a computer overlapping I/O and cpu activity reduces the amount of time disks are idle and increases throughput
2. Interleaved execution offer short transaction with a long transaction usually allows the short transaction to complete quickly
3. In serial execution a short transaction could get stuck behind a long transaction leading to understandable delays in response time or average time taken to complete a transaction

Serializability

- A Serializability Schedule over a set S of committed transactions is a Scheduled whose effect on any consistent database instance is guaranteed to be identical to that of some complete Serial Schedule over S. The database that results after execution the given Schedule is identical to the database instance that results from executing the transactions in some serial order
- As an example the Schedule shown in the Figure below is Serailisble

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)
	COMMIT
COMMIT	

Even though the action of T1 and T2 the result of the Schedule is equivalent T1 running T2