Assignment NO.5

Q.1) What is cascadeless schedule? Why is cascadelessness of schedule desirable?

Sol^n: → Even if a schedule is recoverable, to recover correctly from the failure of a transaction T_i, we may have to roll back several transactions. Such situations occur if transactions have read data written by T_i.

As an illustration, consider the partial schedule of fig.

T_{10}	T_{11}	T_{12}
read (A)		
read (B)		
write (A)		
	read (A)	
	write (A)	
		read (A)

→ Transaction T_{10} writes a value of A that is read by transaction T_{11}. And Transaction T_{11} writes a value of A that is read by transaction T_{12}.

→ Suppose that at this point, T_{10} fails. T_{10} must be rolled back. Since T_{11} is dependent on T_{10}, T_{11} must be rolled back. Since T_{12} is dependent on T_{11}, T_{12} must be rolled back.

→ This phenomenon, in which a single transaction failure leads to a series of transaction roll backs, is called cascading rollback.

→ Cascading rollback is undesirable, since it leads to the undoing of a significant amount of work.

→ It is desirable to restrict the schedules to those where cascading rollbacks cannot occur. Such schedules are called cascade-less schedules.

→ Formally, a cascadeless schedule is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the read operation of T_j. It is easy to verify that every cascadeless schedule is also recoverable.

Page No.
Date

Q.2) Describe the ACID properties. Explain the Usefulness of each.

→ ACID properties are as follows:-

- Atomicity:- Either all operations of the transaction are reflected properly in the database, or none are.

- Consistency:- Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database.

- Isolation:- Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished exection before T_i started or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.

- Durability:- After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

* Usefulness of each properties:-

- Ensuring that this requirement is met is difficult since more changes to the database may still be stored only in the main-memory variables of the transaction, while others may have been written to the database and stored on disk. This "all-or-none" property is referred to as atomicity.

- Transactions are expected to go beyond that to ensure preservation of those application-dependent consistency constraints that are too complex to state using the SQL constructs for data integrity. How this is done is the responsibility of the Programmer who codes a transaction. This property is referred to as consistency.

- Therefore the database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements. This property is refferred to as isolation.

- Even if the system ensures correct execution of a transaction, this serves little purpose if the system subsequently crashes and as a result the system "forgets" about the transaction. Thus, a transaction's actions must persist across crashes. This property is referred to as durability.

Page No
Date

Q.3) Explain the shadow copy technique for atomicity and durability property.

Sol^n:→ The shadow copy scheme is a simple, but extremely inefficient. This scheme, which is based on making copies of the database, called shadow copies, assumes that only one transaction is active at a time.

→ The scheme also assumes that the database is simply a file on disk. A pointer called db-pointer is maintained on disk; it points to the current copy of the database.

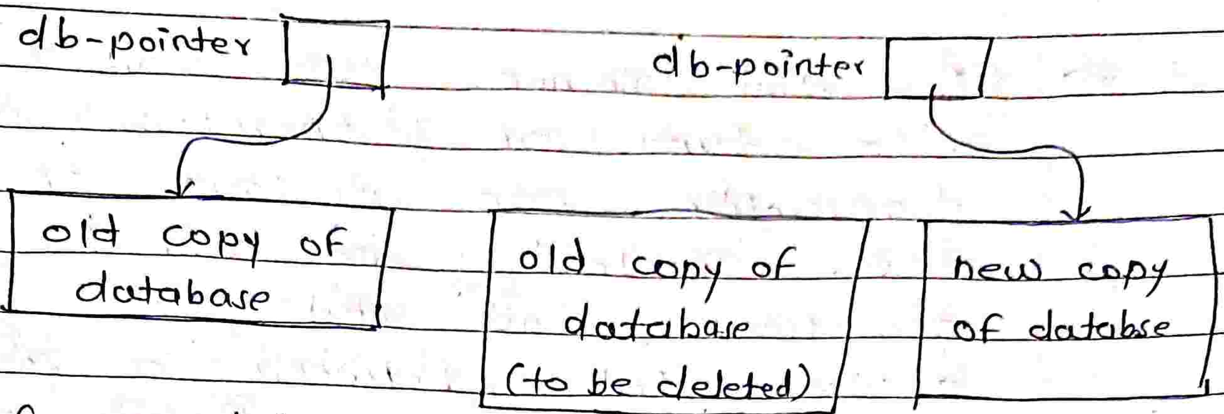
→ In the shadow-copy scheme, a transaction that wants to update the database first creates a complete copy of the database. All updates are done on the new database copy, leaving the original copy, the shadow copy, untouched.

→ If at any point the dransaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.

→ If the transaction completes, it is committed as follows.
First, the OS is asked to make sure that all pages of the new copy of the database have been written out to disk.

→ After the OS has written all the pages to disk, the database system updates the pointer db-pointer to point to the new copy of the database; the new copy then becomes the current copy of the database. The old copy of the database is then deleted.

→ The transaction is said to have been committed at the point where the updated is written to disk depicts the scheme, showing the database, state before and after the update.



(a) Before update

(b) After update.

Fig. Shadow-Copy Technique For Atomicity & Durability.

Q.7) What is serializability? Explain the distinction between serial schedule & serializable schedule.

Soln:→ The data base system must control concurrent execution of transcation, to ensure that the database state remains consistent. Before we examine how the database sure consistency and which schedules will not. Since transactions are programs, it is computationally difficult to determine exactly what operations a transaction performs and how operations of various transactions interact.

→ For this reason, we shall not interpret the type of operations that a transaction can perform on a data item. Instead, we consider only two operations: read and write. We thus assume that, between a read (Q) instruction and a write (Q) instruction on a data item Q, a transaction may perform an arbitrary sequence of operations on the copy of Q that is residing in the local buffer of the transaction.

T1	T2
read (A)	
A:= A - 50	
write (A)	
	read (A)
	temp := A * 0.1
	A:= A - temp
	write (A)
read (B)	
B:= B + 50	
write(B)	
	read(B)
	B:= B + temp
	write (B)

Fig. Concurrent Transactions.

T1	T2
read (A)	
write(A)	
	read (A)
	write (A)
read(B)	
write(B)	
	read (B)
	write(B)

Fig. showing only the read and write instructions.

→ Thus, the only significant operations of a transaction, from a scheduling point of view, are its read and write instructions.

Q.5) What is recoverable schedule? Why is recoverability of schedules desirable?

→

T8	T9
read (A)	
write (A)	
	read (A)
read(B)	

Fig. schedule-1

→ consider schedule-1 ~~a~~in fig. in which T9 is a transaction that performs only one instruction : read(A).

→ Suppose that the system allows T9 to commit immediately after executing the read (A) instruction. Thus, T9 commits before T8 does

→ Now suppose that T8 fails before it commits. Since T9 has read the value of data item A written by T8, we must abort T9 to ensure transaction atomicity.

Date

→ However, T9 has already committed and cannot be aborted. Thus, we have a situation where it is impossible to recover correctly from the failure of T8.

→ Schedule-1, with the commit happening immediately after the read(A) instruction, is an example of a non-recoverable schedule, which should not be allowed. Most database system require that all schedules be recoverable.

→ A recoverable schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the commit operation of Tj.

xox