PART A

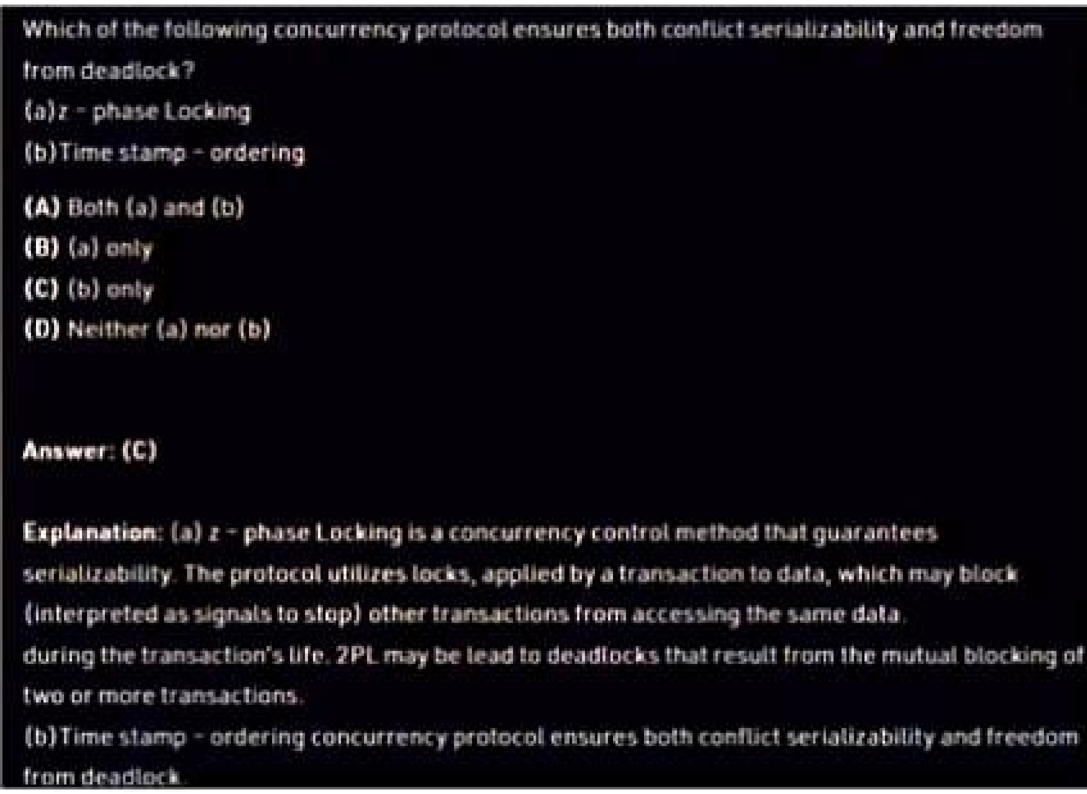
1. Consider the following transactions with data items P and Q initialised to zero:

T1: read(P); read(Q); If P = 0 then Q := Q + 1; write(Q);

T2: read(Q); read(P); If Q = 0 then P := P + 1; write(P);

Solve and find non-serial interleaving of T1 and T2 for concurrent execution leads to a serializable schedule or non serializable schedule. Explain.

2. Analize which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?



Which of the following concurrency protocol ensures both conflict serializability and freedom from deadlock?

(a)z - phase Locking

(b)Time stamp - ordering

(A) Both (a) and (b)

(B) (a) only

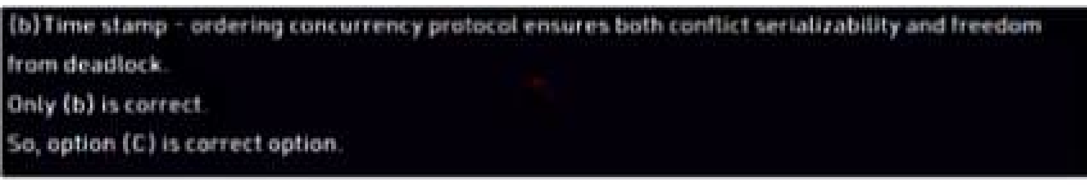
(C) (b) only

(D) Neither (a) nor (b)

Answer: **(C)**

Explanation: (a) z - phase Locking is a concurrency control method that guarantees serializability. The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data. during the transaction's life. 2PL may be lead to deadlocks that result from the mutual blocking of two or more transactions.

(b)Time stamp - ordering concurrency protocol ensures both conflict serializability and freedom from deadlock.



(b)Time stamp - ordering concurrency protocol ensures both conflict serializability and freedom from deadlock.

Only (b) is correct.

So, option (C) is correct option.

- But the process of storing the logs should be done before the actual transaction is applied in the database.

There are two approaches to modify the database:

Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

6. Consider the following actions taken by transaction T1 on database objects X and Y: R(X), W(X), R(Y), W(Y). Give an example of another transaction T2 that, if run concurrently, could interfere with T1.

1. **Explain how the use of strict 2PL would prevent interference between the two transactions.**
2. **Strict 2PL is used in many database systems. Give two reasons for its**

of the following transactions, state the SQL isolation level that you would use and explain why you chose it.

1. A transaction that adds a new part to a supplier's catalog.
2. A transaction that increases the price that a supplier charges for a part.

8. Answer each of the following questions briefly. The questions are based on the following relational schema: Emp(eid: integer, ename: string, age: integer, salary: real, did: integer), Dept(did: integer, dname: string, floor: integer) and on the following update command:

- Replace (salary = 1.1 * EMP salary) where EMP.name = Santa

1. Give an example of a query that would conflict with this command (in a concurrency control sense) if both were run at the same time.
2. Explain what could go wrong, and how locking tuples would solve the problem.
3. Give an example of a query or a command that would conflict with this command, such that the conflict could not be resolved by just locking individual tuples or pages but requires index locking.

9. Suppose that we have only two types of transactions, T1 and T2. Transactions preserve database consistency when run individually. We have defined several integrity constraints such that the DBMS never brings the database into an inconsistent state. Assume that the DBMS does not perform any concurrency control. Give an example schedule of two transactions T1 and T2 that satisfies all these conditions, yet produces a database instance that is not the result of any serial execution of T1 and T2.

ANALYSIS PHASE :-

- The dirty page table has been formed by analysing the pages from the buffer and also a set of active transactions has been identified. When the system encounters crash, ARIES recovery manager starts the analysis phone by finding the last checkpoint log record after that, it prepares dirty pages tables this phone mainly prepares a set of active transactions that are needed to be undo analysis phone, after getting the last checkpoint log record, log record is scanned in forward direction, & update of the set of active transactions, transaction value, & dirty page table are done in the following manner :-
- The 1st recovery manager finds any transaction which is not in the active transaction set, then adds that transaction in that set if it finds an end log record, then that record has been deleted from the transaction table.
- If it finds a log record that describes an update operation, the log record has been added to the dirty page table.

REDO PHASE:-

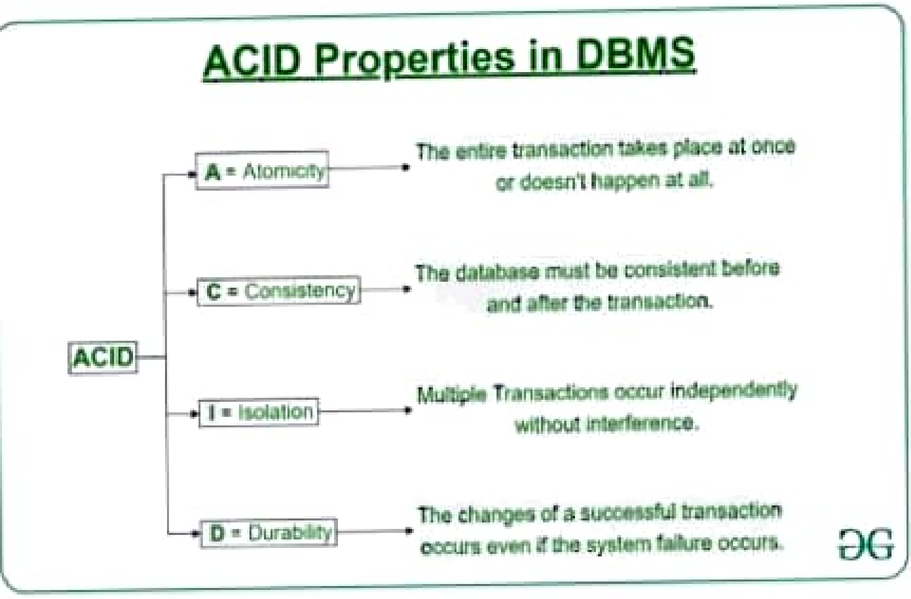
- Redo phase is the second phase where all the transactions that are needed to be executed again take place.
- It executes those operations whose results are not reflected in the disk.
- It can be done by finding the smallest LSN of all the dirty page in dirty page table that defines the log positions, & the Redo operation will start from this position
- This position indicates that either the changes that are made earlier are I the main memory or they have already been flaunted to the disk.
- Thus, for each change recorded in the log, the Redo phase determines whether or not the operations have been re-executed.

UNDO PHASE:-

ART B

ram(1-10)

Explain **ACID properties and illustrate them through examples.**



The entire transaction takes place at once or doesn't happen at all.

The database must be consistent before and after the transaction.

Multiple Transactions occur independently without interference.

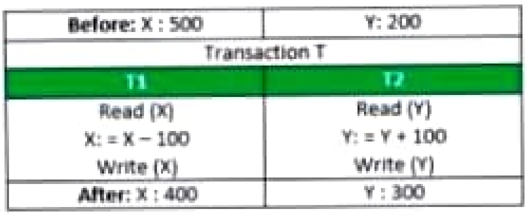
The changes of a successful transaction occurs even if the system failure occurs.

Atomicity By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

- Abort: If a transaction aborts, changes made to the database are not visible.
- Commit: If a transaction commits, changes made are visible.

Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction T consisting of T1 and T2: Transfer of 100 from account X to account Y.



Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

If the transaction fails after completion of T1 but before completion of T2.(say, after write(X) but before write(Y)), then the amount has been deducted from X but not added to Y. This results in an inconsistent database state. Therefore, the transaction must be executed in its entirety in order to ensure the correctness of the database state.

- In the Undo phase, all the transactions that are listed in the active transaction are set here to be undone.
- Thus the log should be scanned background from the end & the recovery manager should Undo the necessary operations.
- Each time an operation is undone, a compensation log recorded has been written to the log.
- This process continues until there is no transaction left in the active transaction set.
- After the successful completion of this phase, the database can resume its normal operations.

Consistency:

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,

The total amount before and after the transaction must be maintained.

Total before T occurs = 500 + 200 = 700.

Total after T occurs = 400 + 300 = 700.

Therefore, the database is consistent. Inconsistency occurs in case T1 completes but T2 fails. As a result, T is incomplete.

Isolation:

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved if these were executed serially in some order.

Let X= 500, Y = 500.

Consider two transactions T and T".

T	T"
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write (Y)	

This results in database inconsistency, due to a loss of 50 units. Hence, ransactions must take place in isolation and changes should be visible only after they have been made to the main memory

Durability:

This property ensures that once the transaction has completed execution, the pdates and modifications to the database are stored in and written to disk nd they persist even if a system failure occurs. These updates now become ermanent and are stored in non-volatile memory. The effects of the ransaction, thus, are never lost.

oint to the updated shadow copy after the transaction reaches partial commit nd all updated pages have been flushed to disk. Db pointer always points to he current consistent copy of the database. In case a transaction fails, old onsistent copy pointed to by the db pointer can be used, and the shadow opy can be deleted. The shadow-database scheme:

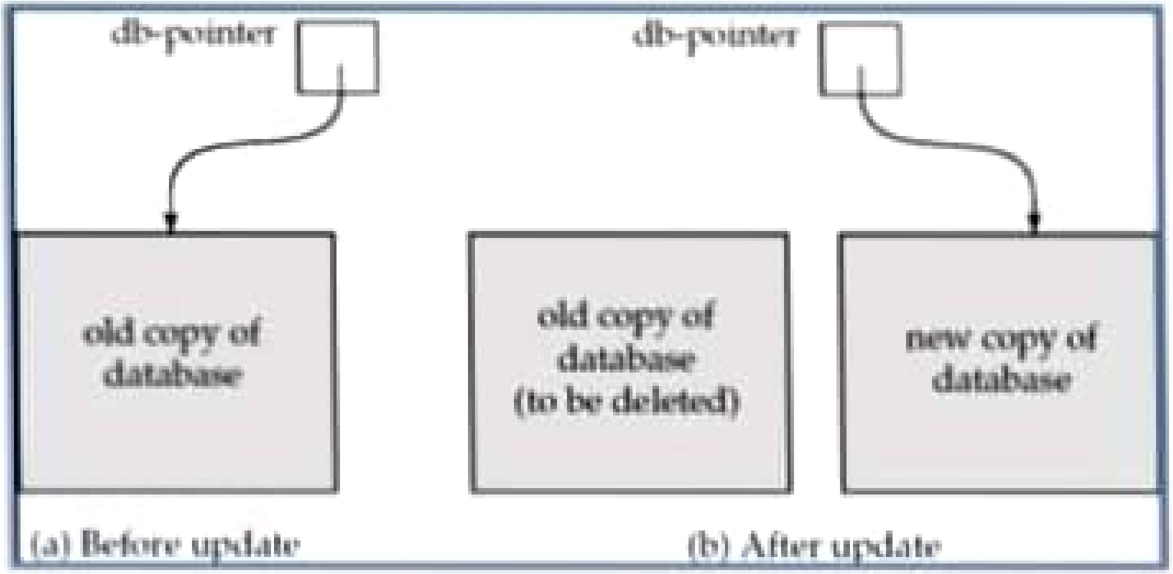
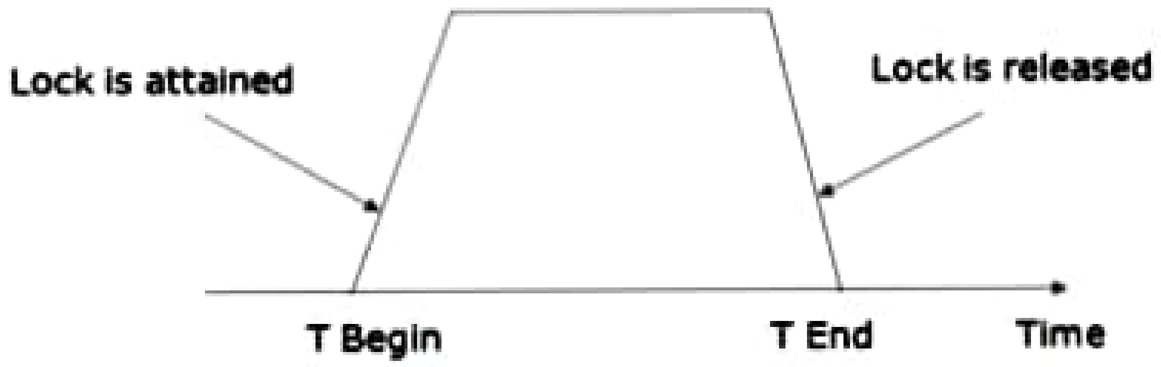


FIGURE 4.2: shadow database

Assumes that only one transaction is active at a time. Assumes disks do not fail Useful for text editors, but extremely inefficient for large databases (why?) Variant called shadow paging reduces copying of data, but is still not practical for large databases does not handle concurrent transactions

DBMS Lock-Based Protocol



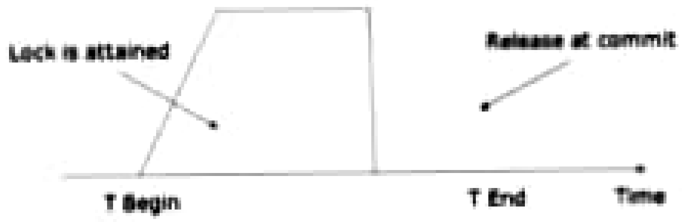
There are two phases of 2PL:

Growing phase: In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking phase: In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

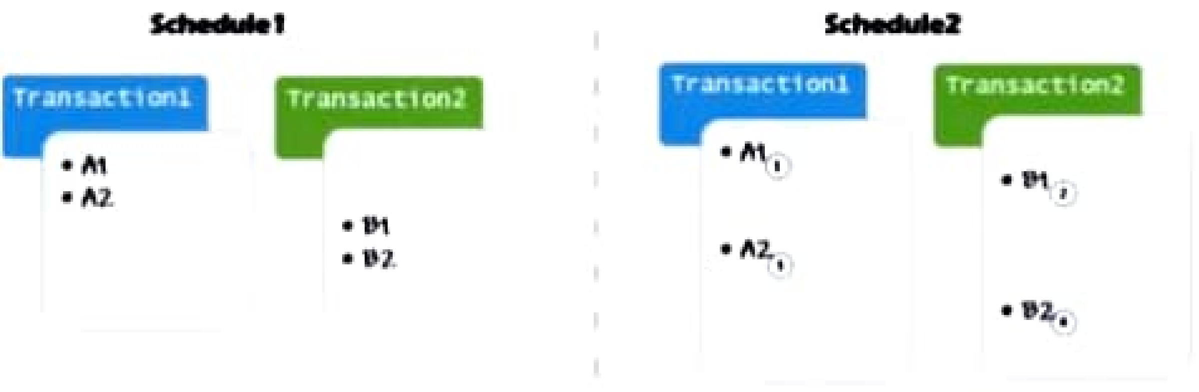
Strict Two-phase locking (Strict-2PL)

- The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.
- Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
- Strict-2PL protocol does not have a shrinking phase of lock release.



Serial schedule both by definition and execution means that the transactions bestowed upon it will take place serially, that is, one after the other. This leaves no place for inconsistency within the database. But, when a set of transactions are scheduled non-serially, they are interleaved leading to the problem of concurrency within the database. Non-serial schedules do not wait for one transaction to complete for the other one to begin. Serializability in DBMS decides if an interleaved non-serial schedule is serializable or not.

EXAMPLE Consider 2 schedules, Schedule1 and Schedule2:



1.2.3.4 – Order of transaction for the Schedule 2 that portrays an interleaved execution.

Schedule1 is a serial schedule consisting of Transaction1 and Transaction2 wherein the operations on data item A (A1 and A2) are performed first and later the operations on data item B (B1 and B2) are carried out serially.

Schedule2 is a non-serial schedule consisting of Transaction1 and Transaction2 wherein the operations are interleaved.

Explanation: In the given scenario, Schedule2 is serializable if the output obtained from both Schedule2 and Schedule1 are equivalent to one another. In a nutshell, a transaction within a given non-serial schedule is serializable if its outcome is equivalent to the outcome of the same transaction when executed serially.

- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.
- In a database transaction, the two main operations are READ and WRITE operations. So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, the data may become inconsistent.

example:

The problem occurs when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution) that makes the values of the items incorrect hence making the database inconsistent.

Consider the below diagram where two transactions T_x and T_y, are performed on the same account A where the balance of account A is $300.

Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A - 50	
t_3	—	READ (A)
t_4	—	A = A + 100
t_5	—	—
t_6	WRITE (A)	—
t_7		WRITE (A)

LOST UPDATE PROBLEM

It does not have cascading abort as 2PL does.

6. Discuss timestamp based locking protocols.

Concurrency Control Protocols

1. Lock-Based Protocols
2. Two Phase Locking Protocol
3. Timestamp-Based Protocols
4. Validation-Based Protocols

- Timestamp based Protocol in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialise the execution of concurrent transactions. The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order.

- The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

- Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.
  Advantages:
  1. Schedules are serializable just like 2PL protocols
  2. No waiting for the transaction, which eliminates the possibility of deadlocks!

7. Describe validation - based locking protocols.

Validation based Protocol in DBMS also known as Optimistic Concurrency Control Technique is a method to avoid concurrency in transactions. In this protocol, the local copies of the transaction data are updated rather than the data itself, which results in less interference while execution of the transaction.

- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.

The Validation based Protocol is performed in the following three phases:

1. Read Phase
2. Validation Phase
3. Write Phase

Read Phase

In the Read Phase, the data values from the database can be read by a transaction but the write operation or updates are only applied to the local data copies, not the actual database.

Validation Phase

Multiple Granularity:

- It can be defined as hierarchically breaking up the database into blocks which can be locked.
- The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes it easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

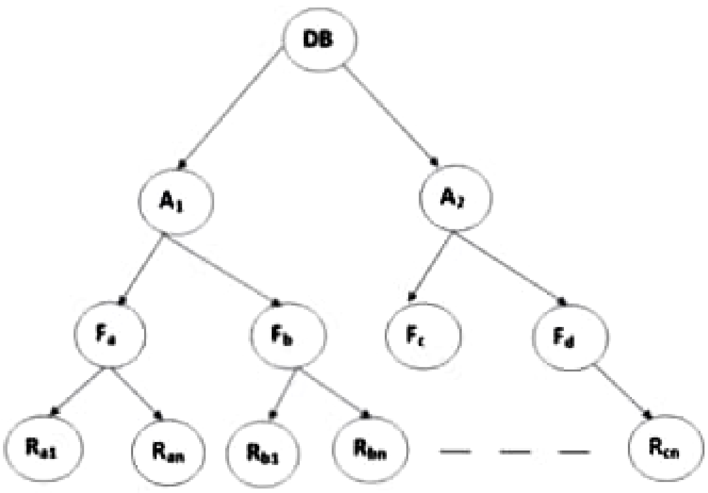


Figure: Multi Granularity tree Hierarchy

The levels of the tree starting from the top level are as follows:

1. Database
2. Area
3. File
4. Record

Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records are represented in more than one file.

- In this example, the highest level shows the entire database. The levels below are file, record, and fields.

9. Explain in detail Storage structure.

A database system provides an ultimate view of the stored data. However, data in the form of bits, bytes get stored in different storage devices.
These storage types differ from one another as per the speed and accessibility. There are the following types of storage devices used for storing the data:

It is the primary area that offers quick access to the stored data. We also know the primary storage as volatile storage. It is because this type of memory does not permanently store the data.

- Main Memory: It is the one that is responsible for operating the data that is available by the storage medium. The main memory handles each instruction of a computer machine. This type of memory is small enough to carry the entire database.
- Cache: It is one of the costly storage media. On the other hand, it is the fastest one. A cache is a tiny storage media which is maintained by the computer hardware usually.

Secondary Storage

Secondary storage is also called Online storage. It is the storage area that allows the user to save and store data permanently. This type of memory does not lose the data due to any power failure or system crash. That's why we also call it non-volatile storage

- Flash Memory: A flash memory stores data in USB (Universal Serial Bus) keys which are further plugged into the USB slots of a computer system. These USB keys help transfer data to a computer system, but it varies in size limits
- Magnetic Disk Storage: This type of storage media is also known as online storage media. it is used for storing the data for a long time. It is capable of storing an entire database. It is the responsibility of the computer system to make availability of the data from a disk to the main memory for further accessing

Tertiary Storage

It is the storage type that is external from the computer system. It has the slowest speed. But it is capable of storing a large amount of data. It is also known as Offline storage.

- Optical Storage: An optical storage can store megabytes or gigabytes of data.

Storage device hierarchy

10. Discuss Deferred database modification and Immediate database modification.

1. Deferred Update :

It is a technique for the maintenance of the transaction log files of the DBMS. It is also called the NO-UNDO/REDO technique. It is used for the recovery of the transaction failures which occur due to power, memory or OS failures. Whenever any transaction is executed, the updates are not made immediately to the database. They are first recorded on the log file and then those changes are applied once a commit is done. This is called the "Re-doing" process. Once the rollback is done none of the changes are applied to the database and the changes in the log file are also discarded. If a commit is done before crashing of the system, then after restarting of the system the changes that have been recorded in the log file are thus applied to the database.

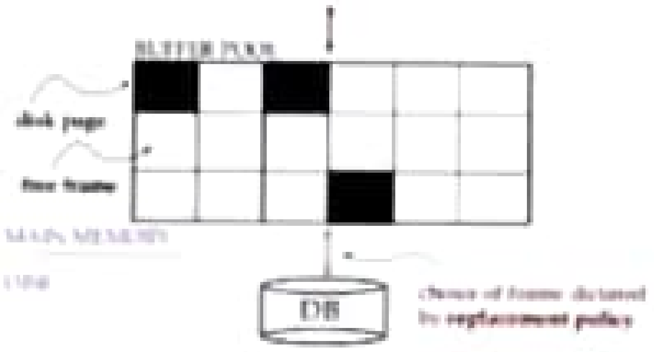
2. Immediate Update :

It is a technique for the maintenance of the transaction log files of the DBMS.It is also called UNDO/REDO technique. It is used for the recovery of the transaction failures which occur due to power, memory or OS failures. Whenever any transaction is executed, the updates are made directly to the database and the log file is also maintained which contains both old and new values. Once a commit is done, all the changes get stored permanently into the database and records in log file are thus discarded.Once rollback is done the old values get restored in the database and all the changes made to the database are also discarded. This is called the "Un-doing" process. If a commit is done before crashing of the system, then after restarting of the system the changes are stored permanently in the database.

12. Explain Buffer Management with a neat diagram.

- A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
- If a user requests a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
- If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
- If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
- The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.

- If the user requests such thrown-out blocks, the buffer manager reads the requested block from the disk to the buffer and then passes the address of the requested block to the user in the main memory.
- However, the internal actions of the buffer manager are not visible to the programs that may create any problem in disk-block requests. The buffer manager is just like a virtual machine.



[illegible]

Concurrency control means that multiple transactions can be executed at the same time and then the interleaved logs occur. But there may be changes in transaction results so maintain the order of execution of those transactions.
During recovery, it would be very difficult for the recovery system to backtrack all the logs and then start recovering.
Recovery with concurrent transactions can be done in the following four ways:

- Interaction with concurrency control
- Transaction rollback
- Checkpoints
- Restart recovery

Interaction with concurrency control :

In this scheme, the recovery scheme depends greatly on the concurrency control scheme that is used. So, to rollback a failed transaction, we must undo the updates performed by the transaction.

Transaction rollback :

In this scheme, we rollback a failed transaction by using the log.
The system scans the log backward for a failed transaction, for every log record found in the log the system restores the data item.

- Whenever more than one transaction is being executed, then the interleaving of logs occurs. During recovery, it would become difficult for the recovery system to backtrack all logs and then start recovering.
- To ease this situation, 'checkpoint' concept is used by most DBMS.

STILL PENDING HERE

12. Explain Buffer Management with a neat diagram.

- A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
- If a user requests a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
- If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
- If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
- The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.

14. Write in detail about Remote Backup Systems.

Remote backup systems provide high availability by allowing transaction processing to continue even if the primary site is destroyed. Detection of failure: Backup site must detect when primary site has failed . To distinguish primary site failure from link failure, maintain several communication links between the primary and the remote backup.

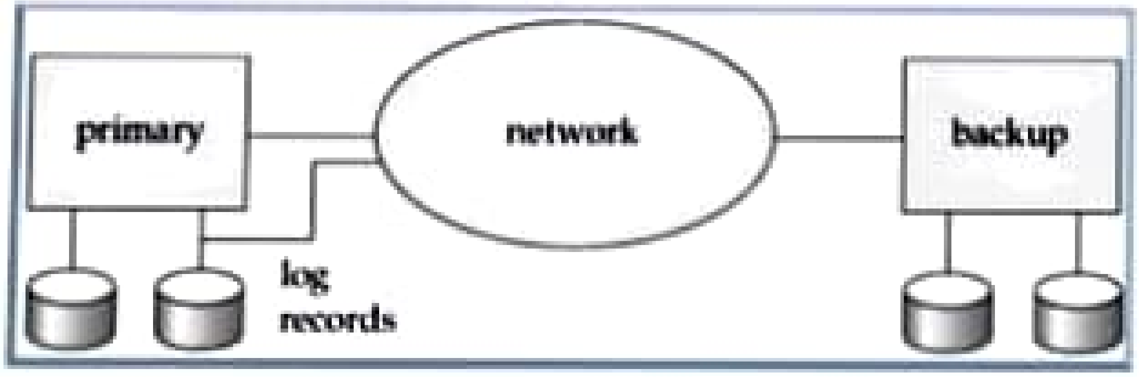


FIGURE 5.13: Remote Backup Systems

ransfer of control:

take over control, the backup site first performs recovery using its copy of e database and all the long records it has received from the primary. Thus, mpleted transactions are redone and incomplete transactions are rolled ck. When the backup site takes over processing it becomes the new primary transfer control back to the old primary. When it recovers, old primary must

1. Deferred database modification:

   The deferred modification technique occurs if the transaction does not modify the database until it has committed.

   In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.
2. Immediate database modification:

   The Immediate modification technique occurs if database modification occurs while the transaction is still active.
3. In this technique, the database is modified immediately after every operation. It follows an actual database modification.

Recovery using Log records

When the system crashes, then the system consults the log to find which transactions need to be undone and which need to be redone.

- If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.

- If log contains record<Tn, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

16. When a transaction is rolled back under timestamp ordering, it is assigned a new timestamp. Why can it not simply keep its old timestamps?

A transaction has rolled back means some other transaction has made the changes in the data which it was supposed to do. Now if it returns with the same timestamp then it will rollback again for the same previous reason and this will continue endlessly hence it is assigned a new timestamp value. (It means that to maintain sequential execution behaviour it will allocate a new timestamp.)

17. Consider the following schedule S1.

- **S1 = 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)**

Check whether S1 is serializable or not. If it is serializable, write its equivalent serial schedule.

18. With a neat diagram explaining NO-UNDO / NO-REDO recovery mechanism in transaction processing.

Time to recover: To reduce delay in takeover, the backup site periodically processes the Redo log records (in effect, performing recovery from previous database state), performs a checkpoint, and can then delete earlier parts of the log. Hot-Spare configuration permits very fast takeover: Backup continually processes redo log records as they arrive, applying the updates locally. When failure of the primary is detected the Backup rolls back incomplete transactions, and is ready to process new transactions. Alternative to remote backup: distributed database with replicated data .Remote backup is faster and cheaper, but less tolerant to failure. Ensure durability of updates by delaying transaction commit until update is logged at backup; avoid this delay by permitting lower degrees of durability. One-safe:
commit as soon as transaction commit log record is written at primary Problem: updates may not arrive at backup before it takes over

15. Explain the Check point log based recovery scheme for recovering the database.

Log-Based Recovery

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Serializable:

This is used to maintain the consistency of the database. It is mainly used in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not. On the other hand, a serial schedule does not need the serializability because it follows a transaction only when the previous transaction is complete. The non-serial schedule is said to be in a serializable schedule only when it is equivalent to the serial schedules, for an n number of transactions. Since concurrency is allowed in this case thus, multiple

transactions can execute concurrently. A serializable schedule helps in improving both resource utilisation and CPU throughput.

- Conflict
- view

Non-Serializable:

The non-serializable schedule is divided into two types, Recoverable and Non-recoverable Schedule.

- Recoverable Schedule
- Non-Recoverable Schedule

20. Suppose that there is a database system that never fails. Analyse whether a recovery management required for this system (REPEATED)

Refer 4th question in part A

PART - C

1. Define a Transaction. List the properties of the transaction.

A transaction is a unit of program execution that accesses and possibl
updates various data items.

Example transaction to transfer $50 from account A to account B:

1. read(A)
2. A:=A-50
3. write(A)
4. read(B)
5. B:=B+50
6. write(B)

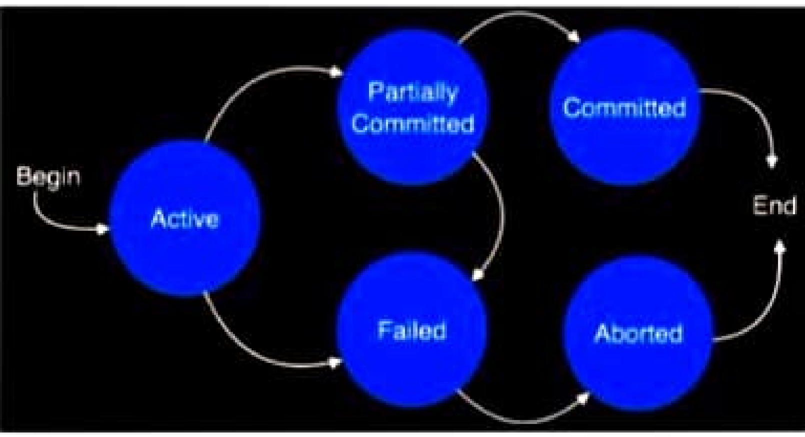
Properties of Transaction:

- Atomicity
- Consistency
- Durability

- Isolation

So as to ensure accuracy, completeness and data integrity.

2. Discuss different phases of the transaction.



Transaction States are as follows:

- Active: In the initial state, the transaction stays in this state while it is executing.
- Partially committed: After the execution of transaction's final operation, it is said to be in a partially committed state.
- Failed: A transaction is said to be in final state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- Aborted: If a transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state. The database recovery mode can select one of the two operations after abort:
    - Re-start
    - Kill

A transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to roll back the failed transaction. But some other transactions may also have used values produced by the failed transaction. So we have to roll back those transactions as well.

There are three types of recoverable schedules

- Cascading Scheduling
- Cascading less Scheduling
- Strict Scheduling

4. Discuss cascade less schedules.

5. Define two phase commit protocol.

6. Demonstrate the implementation of Isolation.

7. Discuss the procedure to test Serializability.

8. List different types of locks and write about compatibility among them.

9. Discuss about Failure Classification.

10. Define a checkpoint.

11. Discuss the failures that can occur with loss of Non - Volatile storage.

12. Demonstrate conflict Serializability.

13. Discuss view Serializability.

14. Explain the distinction between serializable schedule with examples

15. How is the consistency of a transaction preserved?