MODULE IV

TRANSACTION MANAGEMENT

1 Consider the following transactions with data items P and Q initialized 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 any non-serial interleaving of T1 and T2 for concurrent execution leads to a serializable schedule or non serializable schedule. Explain
A)
Two or more actions are said to be in conflict if:

1) The actions belong to different transactions.

2) At least one of the actions is a write operation.

3) The actions access the same object (read or write).

The schedules S1 and S2 are said to be conflict-equivalent if the following conditions are satisfied:

1) Both schedules S1 and S2 involve the same set of transactions (including ordering of actions within each transaction).

2) The order of each pair of conflicting actions in S1 and S2 are the same.

A schedule is said to be conflict-serializable when the schedule is conflict-equivalent to one or more serial schedules.

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

(a)z – phase Locking

(b)Time stamp – ordering

A)

(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.

Only (b) is correct.

3 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 executes any SQL statement that brings the database into an inconsistent state. Assume that the DBMS does not perform any concurrency control. Give an example schedule of two transactions T 1 and T 2 that satisfies all these conditions, yet produces a database instance that is not the result of any serial execution of T 1 and T 2.

4 Suppose that there is a database system that never fails. Analyze whether a recovery manager required for this system.

A) Even in this case the recovery manager is needed to perform roll-back of aborted transactions.

5 Explain the Immediate database Modification technique for using the Log to Ensure transaction atomicity despite failures?

A) 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.

There are two approaches to modify the database:

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.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

Recovery using Log records

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

1. If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.
2. If log contains record<T_n, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

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 T 2 that, if run concurrently to transaction T without some form of concurrency control, could interfere with T 1.
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 popularity.
A)Strict 2PL would require T2 to obtain an exclusive lock on Y before writing to it. This lock would have to be held until T2 committed or aborted; this would block T1 from reading Y until T2 was finished, but there would be no interference.
2) Strict 2PL is popular for many reasons. One reason is that it ensures only ‘safe’ interleaving of transactions so that transactions are recoverable, avoid cascading aborts, etc. Another reason is that strict 2PL is very simple and easy to implement. The lock manager only needs to provide a lookup for exclusive locks and an atomic locking mechanism (such as with a semaphore).

7) Suppliers(sid: integer, sname:string, address: string)
Parts(pid: integer, pname:string, color: string)
Catalog(sid: integer, pid:integer, cost: real) The Catalog
relation lists the prices charged for parts by Suppliers. For each
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 suppliers catalog.
2. A transaction that increases the price that a supplier charges for a part.
A)

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.ename = 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, T 1 and T2.
Transactions preserve database consistency when run
individually. We have defined several integrity constraints such
that the DBMS never executes any SQL statement that brings
the database into an inconsistent state. Assume that

the DBMS does not perform any concurrency control. Give an example schedule of two transactions T 1 and T 2 that satisfies all these conditions, yet produces a database instance that is not the result of any serial execution of T 1 and T 2.

A)The most advance & most difficult recovery algorithm in ARIES recovery algorithm. The advantage of this recovery technique is that it reduces recovery time as well as overhead of a log.

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 :-
- 1st recovery manager finds, any transaction which is not in the active transaction set, then add 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:-

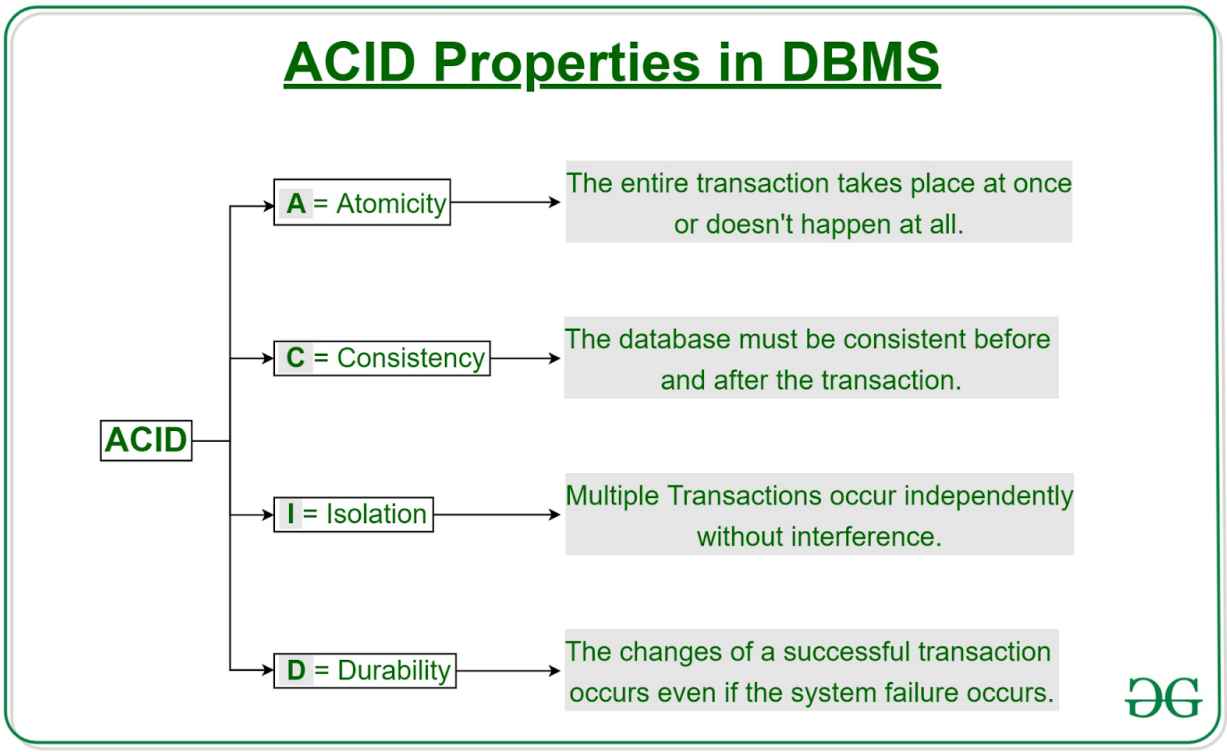
- In the Undo phase, all the transaction, that is listed in the active transaction set here to be undone.

- Thus the log should be scanned background from the end & the recovery manages should Undo the necessary operations.
- Each time an operations 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 competition of this phase, database can resume its normal operations.

PART-B

1 Explain ACID properties and Illustrate them through examples?

A **transaction** is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations. In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.



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’.

Properties
Atomicity - Transaction manager
Consistency - Application programmer
Isolation - Concurrency control manager
Durability - Recovery manager

2 Discuss How do you implement Atomicity and Durability?

The recovery-management element of a database system can support atomicity and durability by a range of plans. This plan, which is based on making copies of the database, called shadow copies, presumes that just one deal is active at a time. In the shadow-copy plan, a deal that desires to upgrade the database first develops a total copy of the database. If at any point the deal has actually to be terminated, the system simply erases the brand-new copy. If the deal finishes, it is dedicated as follows. After the operating system has actually composed all the pages to disk, the database system updates the guideline db-pointer to point to the brand-new copy of the database; the brand-new copy then ends up being the present copy of the database.

Atomicity handle these failures:

- User terminates deal (e.g., cancel button).

- System terminates deal (e.g., deadlock).

-Transaction terminates itself (e.g., unanticipated db state).

- System crashes.

Durability handle this kind of failure - Media failure.

Implementation

Atomicity can be specified as a deal including 2 or more discrete pieces of info, either all of the pieces are dedicated or none are. The recovery-management element of a database system carries out the assistance for atomicity and durability. The atomicity and durability residential or commercial properties of deals are made sure by the shadow-copy implementation of the recovery-management element.

3 Illustrate Concurrent execution of transaction with examples?

Concurrent Execution of Transaction

In the transaction process, a system usually allows executing more than one transaction simultaneously. This process is called a concurrent execution.There are following different types of problems or conflicts which occur due to concurrent execution of transaction:

1. Lost update problem (Write – Write conflict)
This type of problem occurs when two transactions in database access the same data item and have their operations in an interleaved manner that makes the value of some database item incorrect.
If there are two transactions T1 and T2 accessing the same data item value and then update it, then the second record overwrites the first record.

Example: Let's take the value of A is 100

Time	Transaction T1	Transaction T2
t1	Read(A)	
t2	A=A-50	
t3		Read(A)
t4		A=A+50
t5	Write(A)	
t6		Write(A)

Here,

- At t1 time, T1 transaction reads the value of A i.e., 100.
- At t2 time, T1 transaction deducts the value of A by 50.
- At t3 time, T2 transactions read the value of A i.e., 100.
- At t4 time, T2 transaction adds the value of A by 150.
- At t5 time, T1 transaction writes the value of A data item on the basis of value seen at time t2 i.e., 50.
- At t6 time, T2 transaction writes the value of A based on value seen at time t4 i.e., 150.
- So at time T6, the update of Transaction T1 is lost because Transaction T2 overwrites the value of A without looking at its current value.
- Such type of problem is known as the Lost Update Problem.

Dirty read problem (W-R conflict)

This type of problem occurs when one transaction T1 updates a data item of the database, and then that transaction fails due to some reason, but its updates are accessed by some other transaction.
Example: Let's take the value of A is 100

Time	Transaction T1	Transaction T2
t1	Read(A)	
t2	A=A+20	
t3	Write(A)	
t4		Read(A)
t5		A=A+30
t6		Write(A)
t7	Write(B)	

Here,

- **At t1 time, T1 transaction reads the value of A i.e., 100.**
- **At t2 time, T1 transaction adds the value of A by 20.**
- **At t3 time, T1transaction writes the value of A (120) in the database.**
- **At t4 time, T2 transactions read the value of A data item i.e., 120.**
- **At t5 time, T2 transaction adds the value of A data item by 30.**
- **At t6 time, T2transaction writes the value of A (150) in the database.**
- **At t7 time, a T1 transaction fails due to power failure then it is rollback according to atomicity property of transaction (either all or none).**

- **So, transaction T2 at t4 time contains a value which has not been committed in the database. The value read by the transaction T2 is known as a dirty read.**

Unrepeatable read (R-W Conflict)

It is also known as an inconsistent retrieval problem. If a transaction T1 reads a value of data item twice and the data item is changed by another transaction T2 in between the two read operation. Hence T1 access two different values for its two read operation of the same data item.
Example: **Let's take the value of A is 100**

Time	Transaction T1	Transaction T2
t1	Read(A)	
t2		Read(A)
t3		A=A+30
t4		Write(A)
t5	Read(A)	

Here,

- **At t1 time, T1 transaction reads the value of A i.e., 100.**
- **At t2 time, T2transaction reads the value of A i.e., 100.**
- **At t3 time, T2 transaction adds the value of A data item by 30.**
- **At t4 time, T2 transaction writes the value of A (130) in the database.**
- **Transaction T2 updates the value of A. Thus, when another read statement is performed by transaction T1, it accesses the new value of A, which was updated by T2. Such type of conflict is known as R-W conflict.**

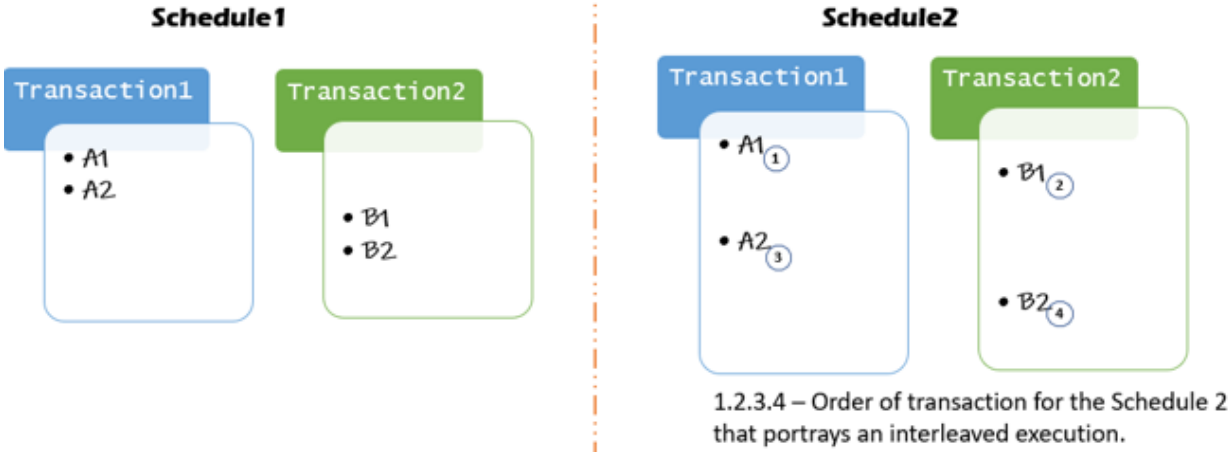
4 Discuss Serializability in detail with an example?

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.

- Serializable means obtaining an equivalent output as of a serial schedule for the same ‘n’ number of transactions.
- Serializability helps preserve the consistency and concurrency of a database.
- There are 2 methods widely used to check serializability i.e. Conflict equivalent and View equivalent.
- As a rule of thumb, it can be stated that all conflict serializable schedules can be view serializable, but all view serializable schedules may or may not be conflict serializable.

Example of Serializability

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.

Types of Serializability

A schedule can be checked for serializability in one of the 3 methods mentioned below:

1. Result Equivalent Schedule

Two schedules, S1 and S2 are said to result equivalent if they produce the same output obtained when the schedules are serially executed.

2. Conflict Equivalent Schedule

When either of a conflict operation such as Read-Write or Write-Read or Write-Write is implemented on the same data item at the same time within different transactions then the schedule holding such transactions is said to be a conflict schedule.Two schedules (one being serial schedule and another being non-serial) are said to be conflict serializable if the conflict operations in both the schedules are executed in the same order.

3. View Equivalent Schedule

Two schedules (one being serial schedule and another being non-serial) are said to be view serializable if they satisfy the rules for being view equivalent to one another.

5 Discuss two phase locking protocol and strict two phase locking protocols?

Two-Phase Locking –

A transaction is said to follow the Two-Phase Locking protocol if Locking and Unlocking can be done in two phases.

1. Growing Phase: New locks on data items may be acquired but none can be released.
2. Shrinking Phase: Existing locks may be released but no new locks can be acquired.

There are two types of Locks available **Shared S(a)** and **Exclusive X(a)**. If lock conversion is allowed, then upgrading of lock(from S(a) to X(a)) is allowed in the Growing Phase, and downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

6 Describe Timestamp based locking protocols?

Timestamp is a unique identifier created by the DBMS to identify a transaction. They are usually assigned in the order in which they are submitted to the system.

The main idea for this protocol is to order the transactions based on their Timestamps. A schedule in which the transactions participate is then serializable and the only *equivalent serial schedule permitted* has the transactions in the order of their Timestamp Values.

An algorithm must ensure that, for each item accessed by *Conflicting Operations* in the schedule, the order in which the item is accessed does not violate the ordering. To ensure this, use two Timestamp Values relating to each database item **X**.

- **W_TS(X)** is the largest timestamp of any transaction that executed **write(X)** successfully.
- **R_TS(X)** is the largest timestamp of any transaction that executed **read(X)** successfully.

Basic Timestamp Ordering –

Every transaction is issued a timestamp based on when it enters the system. Suppose, if an old transaction T_i has timestamp $TS(T_i)$, a new transaction T_j is assigned timestamp $TS(T_j)$ such that $\mathbf{TS(T_i) < TS(T_j)}$.

Whenever some Transaction *T* tries to issue a R_item(X) or a W_item(X), the Basic TO algorithm compares the timestamp of *T* with **R_TS(X) & W_TS(X)** to ensure that the Timestamp order is not violated. This describes the Basic TO protocol in the following two cases.

1. Whenever a Transaction *T* issues a **W_item(X)** operation, check the following conditions:
    - If ***R_TS(X) > TS(T)*** or if ***W_TS(X) > TS(T)***, then abort and rollback T and reject the operation. else,
    - Execute W_item(X) operation of T and set W_TS(X) to TS(T).
2. Whenever a Transaction *T* issues a **R_item(X)** operation, check the following conditions:
    - If ***W_TS(X) > TS(T)***, then abort and reject T and reject the operation, else
    - If W_TS(X) <= TS(T), then execute the R_item(X) operation of T and set R_TS(X) to the larger of TS(T) and current R_TS(X).

7 Describe Validation-based locking protocols?.

Validation Based Protocol is also called Optimistic Concurrency Control Technique. This protocol is used in DBMS (Database Management System) for avoiding concurrency in transactions. It is called optimistic because of the assumption it makes, i.e. very less interference occurs, therefore, there is no need for checking while the transaction is executed.

In this technique, no checking is done while the transaction is been executed. Until the transaction end is reached updates in the transaction are not applied directly to the database. All updates are applied to local copies of data items kept for the transaction. At the end of transaction execution, while execution of the transaction, a **validation phase** checks whether any of transaction updates

violate serializability. If there is no violation of serializability the transaction is committed and the database is updated; or else, the transaction is updated and then restarted.

Validation based protocol is useful for rare conflicts. Since only local copies of data are included in rollbacks, cascading rollbacks are avoided. This method is not favourable for longer transactions because they are more likely to have conflicts and might be repeatedly rolled back due to conflicts with short transactions.

In order to perform the Validation test, each transaction should go through the various phases as described above. Then, we must know about the following three time-stamps that we assigned to transaction T_i, to check its validity:

1. Start(T_i): It is the time when T_i started its execution.

2. Validation(T_i): It is the time when T_i just finished its read phase and begin its validation phase.

3. Finish(T_i): the time when T_i end it's all writing operations in the database under write-phase.

Two more terms that we need to know are:

1. Write_set: of a transaction contains all the write operations that T_i performs.

2. Read_set: of a transaction contains all the read operations that T_i performs.

In the Validation phase for transaction T_i the protocol inspect that T_i doesn't overlap or intervene with any other transactions currently in their validation phase or in committed. The validation phase for T_i checks that for all transaction T_j one of the following below conditions must hold to being validated or pass validation phase:

1. Finish(T_j)<Starts(T_i), since T_j finishes its execution means completes its write-phase before T_i started its execution(read-phase). Then the serializability indeed maintained.

2. T_i begins its write phase after T_j completes its write phase, and the read_set of T_i should be disjoint with write_set of T_j.

3. T_j completes its read phase before T_i completes its read phase and both read_set and write_set of T_i are disjoint with the write_set of T_j.

8 Discuss in detail Multiple Granularity?

Granularity – It is the size of the data item allowed to lock.

- 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 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.

For example: Consider a tree which has four levels of nodes.

- The first level or higher level shows the entire database.
- The second level represents a node of type area. The higher level database consists of exactly these areas.
- The area consists of children nodes which are known as files. No file can be present in more than one area.

- Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.
- Hence, the levels of the tree starting from the top level are as follows:
    1. Database
    2. Area
    3. File
    4. Record

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

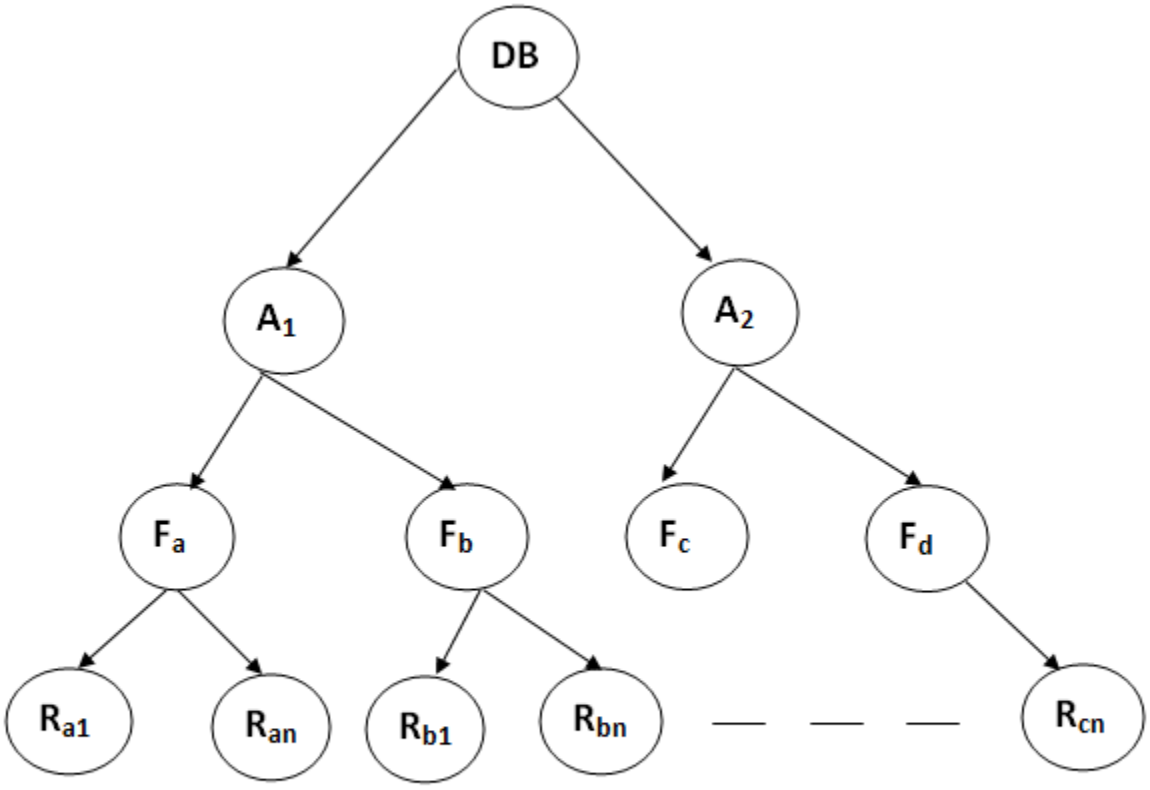


Figure: Multi Granularity tree Hierarchy

9 Explain in detail Storage structure?

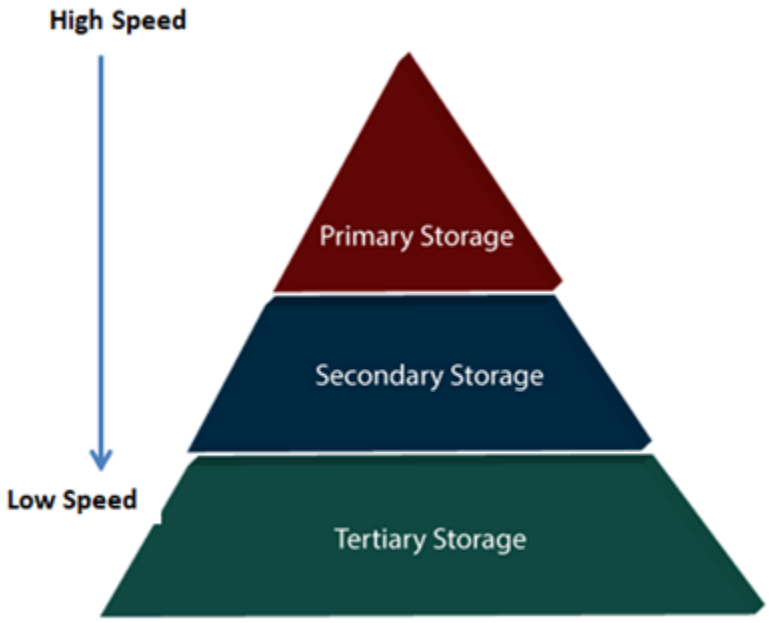
A database system provides an ultimate view of the stored data. However, data in the form of bits, bytes get stored in different storage devices.

Types of Data Storage

For storing the data, there are different types of storage options available.

- Primary Storage
- Secondary Storage
- Tertiary Storage

Primary Storage (**Main Memory,cache**)



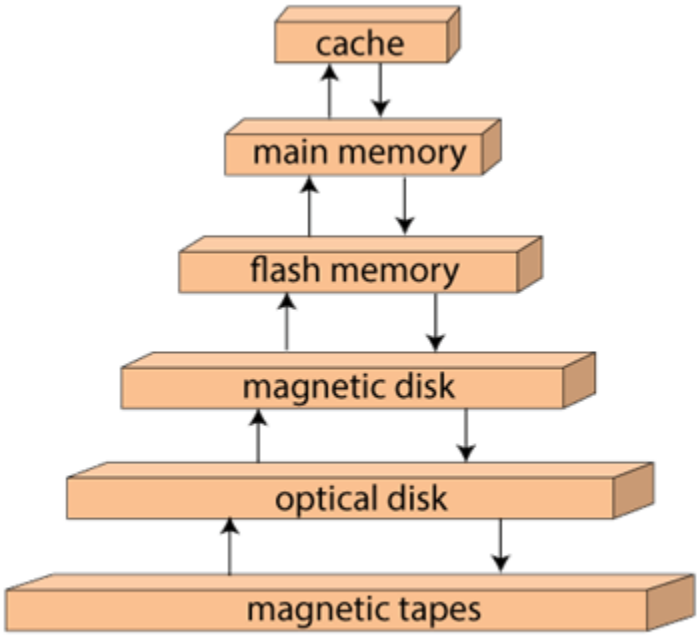
It is the primary area that offers quick access to the stored data. We also know the primary storage as volatile storage. It is because this type of memory does not permanently store the data.

Secondary Storage(**Flash Memory,Magnetic Disk Storage**)

Secondary storage is also called as 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.

Tertiary Storage(**Optical Storage,Tape 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. Tertiary storage is generally used for data backup.



Storage device hierarchy

10 Discuss Deferred database modification and Immediate database modification?

Deferred Update :

It is a technique for the maintenance of the transaction log files of the DBMS. It is also called NO-UNDO/REDO technique. It is used for the recovery of the transaction failures which occur due to power, memory or OS failures.

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.

S.NO.	Deferred Update	Immediate Update
1.	In deferred update, the changes are not applied immediately to the database.	In immediate update, the changes are applied directly to the database.
2.	The log file contains all the changes that are to be applied to the database.	The log file contains both old as well as new values.
3.	In this method once rollback is done all the records of log file are discarded and no changes are applied to the database.	In this method once rollback is done the old values are restored into the database using the records of the log file.
4.	Concepts of buffering and caching are used in deferred update method.	Concept of shadow paging is used in immediate update method.
5.	The major disadvantage of this method is that it requires a lot of time for recovery in case of system failure.	The major disadvantage of this method is that there are frequent I/O operations while the transaction is active.

11 Discuss how you recover from Concurrent transactions?

Concurrency Control deals with interleaved execution of more than one transaction.Recovery with concurrent transactions can be done in the following four ways.

1. Interaction with concurrency control
2. Transaction rollback
3. Checkpoints
4. 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 a failed transaction, for every log record found in the log the system restores the data item.

Checkpoints :

- Checkpoints is a process of saving a snapshot of the applications state so that it can restart from that point in case of failure.
- we used checkpoints to reduce the number of log records that the system must scan when it recovers from a crash.

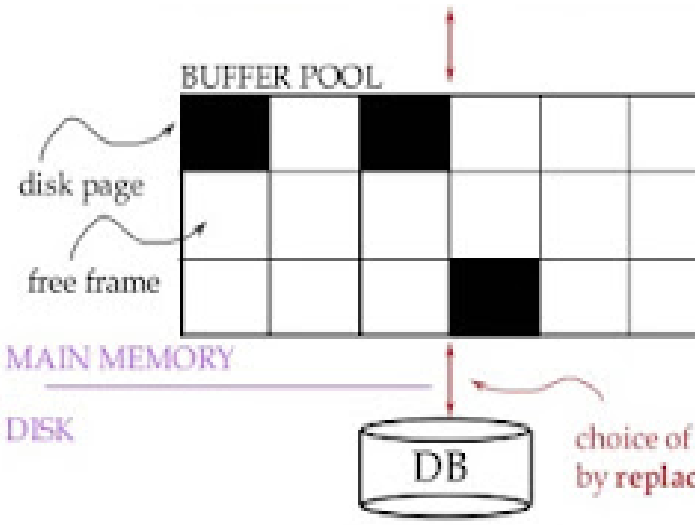
- In a concurrent transaction processing system, we require that the checkpoint log record be of the form <checkpoint L>, where **'L'** is a list of transactions active at the time of the checkpoint.

Restart recovery :

- When the system recovers from a crash, it constructs two lists.
- The undo-list consists of transactions to be undone, and the redo-list consists of transaction to be redone.
- The system constructs the two lists as follows: Initially, they are both empty. The system scans the log backward, examining each record, until it finds the first <checkpoint> record.

12 Explain Buffer Management with a neat diagram?

The **buffer manager** is the software layer that is responsible for bringing pages from physical disk to main memory as needed. The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as **buffer pool.** The main memory pages in the buffer pool are called **frames.**



Buffer Management in a DBMS

(Page Requests from Higher Level)

- **Data must be in RAM for DBMS to operate on it!**
- **Buffer manager hides the fact that not all data is in RAM.**

The goal of the buffer manager is to ensure that the data requests made by programs are satisfied by copying data from secondary storage devices into buffer. Infact, if a program performs reading from existing buffers. similarly, if a program performs an output statement: it calls the buffer manager for

output operation - to satisfy the requests by writing to the buffers. Therefore, we can say that input and output operation occurs between the program and the buffer area only.

13 Explain different types of Advanced Recovery Techniques.

14 Write in detail about Remote Backup systems?

Remote backup systems provide a wide range of availability, allowing the transaction processing to continue even if the primary site is destroyed by a fire, flood or earthquake.

Data and log records from a primary site are continuously backed up into a remote backup site.

One can achieve ‘wide range availability’ of data by performing transaction processing at one site, called the ‘**primary site**’, and having a ‘**remote backup**’ site where all the data from the primary site are duplicated.The remote site is also called ‘**secondary site**’.

The remote site must be **synchronized** with the primary site, as updates are performed at the primary.

In designing a remote backup system, the following points are important.

a) Detection of failure: It is important for the remote backup system to detect when the primary has failed.

b) Transfer of control: When the primary site fails, the backup site takes over the processing and becomes the new primary site.

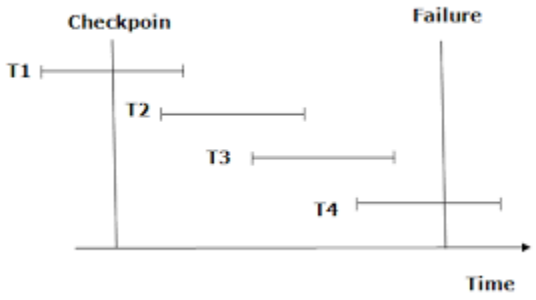
c) Time to recover: If the log at the remote backup becomes large, recovery will take a long time.

d) Time to commit: To ensure that the updates of a committed transaction are durable, a transaction should not be announced committed until its log records have reached the backup site.

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

- The checkpoint is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.
- The checkpoint is like a bookmark. While the execution of the transaction, such checkpoints are marked, and the transaction is executed then using the steps of

the transaction, the log files will be created.



- When it reaches to the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file. Then the log file is updated with the new step of transaction till next checkpoint and so on.
- The checkpoint is used to declare a point before which the DBMS was in the consistent state, and all transactions were committed
- The recovery system reads log files from the end to start. It reads log files from T4 to T1.
- Recovery system maintains two lists, a redo-list, and an undo-list.
- The transaction is put into redo state if the recovery system sees a log with <Tn, Start> and <Tn, Commit> or just <Tn, Commit>. In the redo-list and their previous list, all the transactions are removed and then redone before saving their logs.

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

If write_TS(X) >TS(T), then do not execute the write operation but continue processing.This is because some

transaction with timestamp greater than TS(T)—and hence after T in the timestamp ordering—has already written the value of X. Hence, we must ignore the write_item(X) operation of T because it is already outdated and obsolete.

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.

Explanation: For conflict serializability of a schedule(which gives same effect as a serial schedule) we should check for conflict operations, which are Read-Write, Write-Read and Write-Write between each pair of transactions, and based on those conflicts we make a precedence graph, if the graph contains a cycle, it's not a conflict serializable schedule.

To make a precedence graph: if Read(X) in Ti followed by Write(X) in Tj (hence a conflict), then we draw an edge from Ti to Tj (Ti -> Tj)

If we make a precedence graph for S1 and S2 , we would get directed edges for S1 as T2->T1, T2->T3, T3->T1, and for S2 as T2->T1, T2->T3, T3->T1, T1->T2. In S1 there is no cycle, but S2 has a cycle. Hence only S1 is conflict serializable.

Note : The serial order for S1 is T2 -> T3 -> T1.

18 With a neat diagram explain NO-UNDO/NO-REDO recovery mechanism in transaction processing?

During transaction execution, the updates are recorded only in the log and in the cache buffers. After the transaction reaches its commit point and the log is force-written to disk, the updates are recorded in the database. If a transaction fails before reaching its commit point, there is no need to undo any operations because the transaction has not affected the database on disk in any way. Therefore, only REDO-type log entries are needed in the log, which include the new value (AFIM) of the item written by a write operation. The UNDO-type log entries are not needed since no undoing of operations will be required during recovery. Although this may sim-plify the recovery process, it cannot be used in practice unless transactions are short and each transaction changes few items.

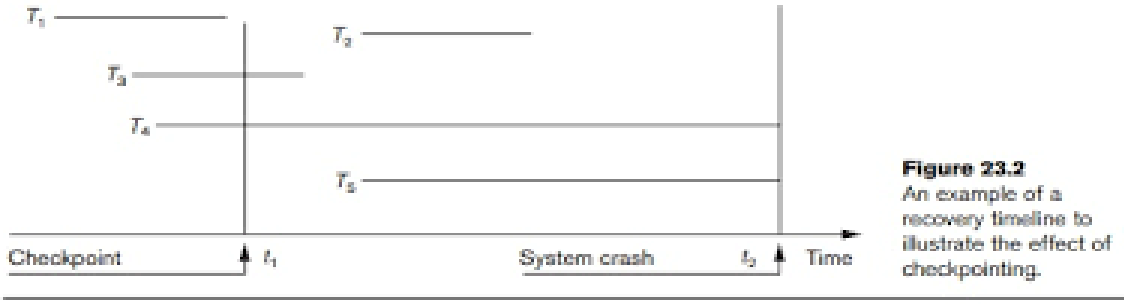


Figure 23.2
An example of a recovery timeline to illustrate the effect of checkpointing.

19 Explain the serializable and non serializable schedule?

Serial Schedule

The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

Non-serial Schedule

- If interleaving of operations is allowed, then there will be non-serial schedule.
- It contains many possible orders in which the system can execute the individual operations of the transactions.

Serializable schedule

- The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.
- It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
- A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

20 Suppose that there is a database system that never fails. Analyze whether a recovery manager required for this system.

Even in this case the recovery manager is needed to perform roll-back of aborted transactions…

Types of schedules in DBMS