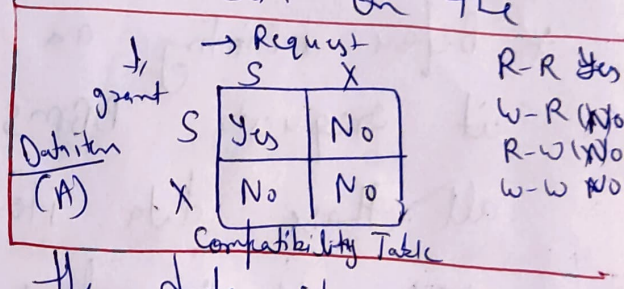
Unit-5

Lock-Based Protocal

In this type of Protocal, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock-

1) Shared lock (S)

* It is also known as 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.



2) Exclusive lock (X)

* 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.

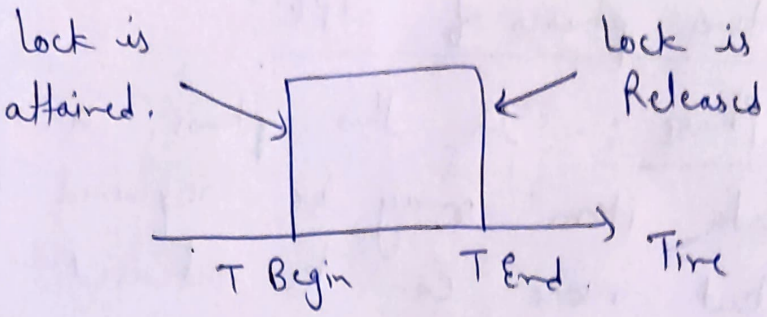
There are four types of lock protocol

1) Simplistic lock protocol

It is the simplest way of locking the data while transaction. Simplistic lock-Based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.

2) Pre-claiming lock protocols

* Pre-claiming lock protocols evaluate the transaction to list all the data items on which they need locks.
* Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items.
* If all the locks are granted then this protocol allows the transaction to begin. When the transaction is completed then it releases all the lock.
* If all the locks are not granted then this protocol allows the transaction to rolls back and waits until all the locks are granted.

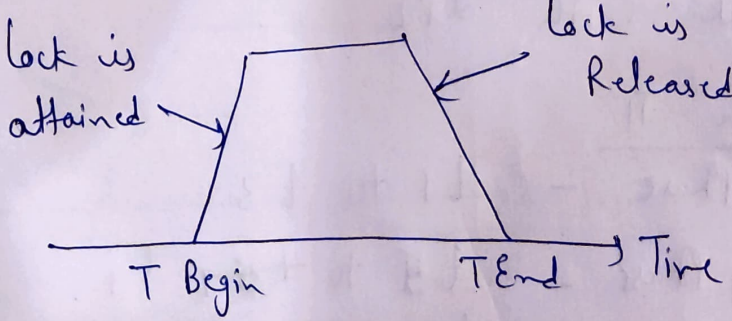


3) Two-Phase Locking (2PL)

* The two-phase locking Protocal divides the execution phase of the transaction into three parts.
* In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
* In the second part, the transaction acquires all the locks.

The third phase is started as soon as the transaction releases its first lock.

* In the third phase, the transaction cannot demand any new locks, It only releases the acquired locks.



There are two phases of 2PL:

* Growing Phase :- In this 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.

Example :-

	T1	T2
t1	lock-S(A)	
t2		lock-S(A)
t3	lock-X(B)	
t4	—	
t5	Unlock(A)	
t6		lock-X(C)
t7	Unlock(B)	
t8		Unlock(A)
t9		Unlock(C)

Lock Point → T1 (t4)

Lock Point → T2 (t6)

On the Basis of lock Point

T1 → T2

Note:- 2PL always provide Serializability.

The following way shows how unlocking and locking work with 2PL

Transaction T1

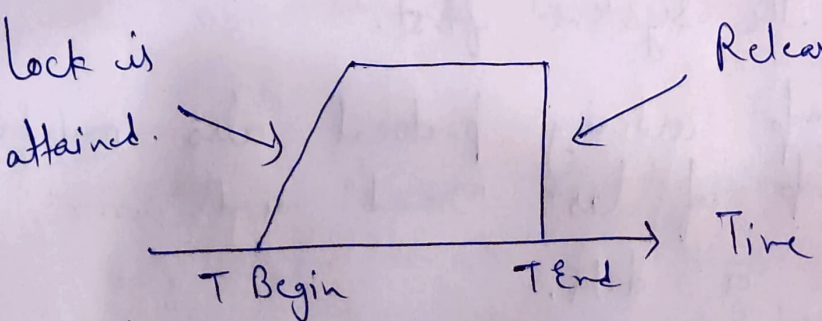
* Growing Phase - t1 to t3
* Shrinking Phase - t5 to t7
* Lock Point - t4

Transaction T2

* Growing Phase - t_2 to t_6
* Shrinking Phase - t_8 to t_9
* lock Point - t_7

4) 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 Shrinking Phase of lock Release.



Time Stamp Ordering Protocal

* The timestamp ordering protocal 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 protocal uses system time or logical counter.
* The lock-based protocal is used to manage the order between conflicting pairs among transactions at the execution time. But timestamp based protocals 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 protocal also maintains the timestamp of last "read" and "write" operation on a data.

Basic Timestamp ordering Protocal work as follows -

1) Check the following condition whenever a transaction T_i issues a Read (x) operation -

* if W_TS (X) > TS (T_i) then the operation is rejected.
* if W_TS (X) <= TS (T_i) then the operation is Executed.
* Timestamps of all the data items are updated.

2) Check the following Condition whenever a transaction T_i issues a Write (X) operation -

* if TS (T_i) < R_TS (X) then the operation is rejected.
* if TS (T_i) < W_TS (X) then the operation is rejected and T_i is rolled back otherwise the operation is executed.

Where:-

TS (T_i) - denotes the timestamp of Transaction T_i.

R_TS (T_i) - denotes the Read time-stamp of data item X. (latest)

W_TS(X):- denotes the write time-stamp of data-item X. (latest)

3) Validation Based Protocol

It is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases -

1) Read Phase: In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.

2) Validation Phase: In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.

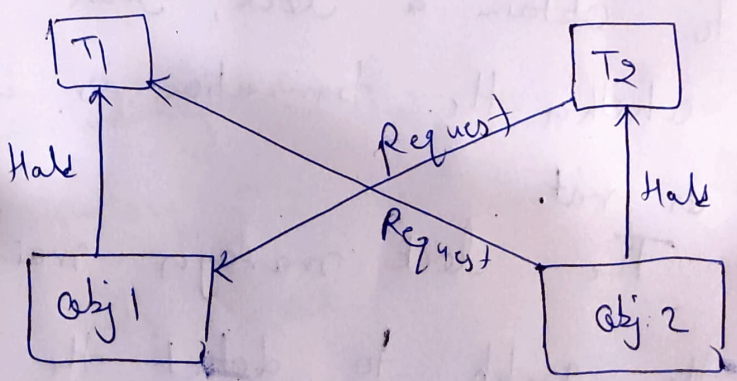
3) Write Phase:- If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Deadlock in DBMS

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks.

As a result, none of the transactions can proceed, leading to a situation where they are stuck in deadlock.

Deadlock can happen in multi-user environments when two or more transactions are running concurrently and try to access the same data in a different order. When this happens, one transaction may hold a lock on a resource that another transaction needs, while the second transaction may hold a lock on a resource that the first transaction needs. Both transaction are then blocked, waiting for the other to release the resource they need.



Deadlock

Deadlock Avoidance

When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restarting the database. This is a waste of time and Resource.

Deadlock Avoidance mechanism is used to detect any deadlock situation in advance.

A method like "wait for graph" is used for detecting the deadlock situation but this method is suitable only for the smaller database.

For larger database, deadlock prevention method can be used.

Deadlock Detection

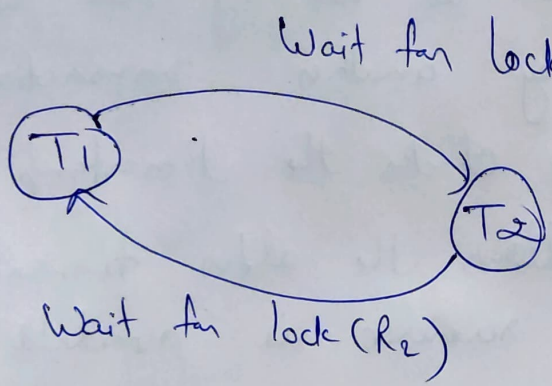
In a database, when a transaction waits indefinitely to obtain a lock, then the DBMS should detect whether the transaction is involved in a deadlock or not.

The lock manager maintains a wait for the graph to detect the deadlock cycle in the database.

Wait for Graph

* This is the suitable method for deadlock detection. In this method, a graph is created based on the transaction and their lock. If the created graph as a cycle or closed loop, then there is a deadlock.

* The wait for the graph is maintained by the system for every transaction which is waiting for some data held by the others. The system keeps checking the graph if there is any ~~of~~ cycle ~~in~~ the graph.



T1 is waiting for T2 to release its lock and similary, T2 is waiting for T1 to release its lock.

Deadlock Prevention

* This method is suitable for a larger database. If the resources are allocated in such a way that deadlock never occurs, then the deadlocks can be prevented.

* The DBMS analyzes the operations of the transaction whether they can create a deadlock situation or not. If they do, then DBMS never allowed that transaction to be executed.

Wait - Die Scheme

In this scheme, if a transaction requests for a resource which is already held with a conflicting lock by another transaction then the DBMS simply checks the timestamp of both transactions. It allows the older transaction to wait until the resource is available for execution.

Wound Wait Scheme

* In this scheme, if the older transaction requests for a resource which is held by the younger transaction, then older transaction forces younger one to kill the transaction and release the

resource. After the minute delay, the younger transaction is restarted but with the same timestamp.

* If the older transaction has held a resource which is requested by the younger transaction, then the younger transaction is asked to wait until older releases it.

Database Failure

Failure in terms of a database can be defined as its inability to execute the specified transaction or loss of data from the database.

A Database failure can be classified as-

1) Transaction failure
2) System Crash
3) Disk Failure

1) Transaction failure

If a transaction is not able to execute or it comes to a point from where the transaction becomes incapable of executing further then it is termed as a failure in a transaction.

Reason for a transaction failure in DBMS:

* Logical Error :- A logical error occurs if a transaction is unable to execute because of some mistakes in the code or due to the presence of some internal faults.

* System Error :- Where the termination of an active transaction is done by the database system itself due to some system issue or because the database management system is unable to proceed with the transaction.
for eg. The system aborts an active transaction, in case of deadlock or resource unavailability.

2) System Crash

It occurs due to power failure or other H/w or S/w failure.
Eg. Operating System Error

3) Disk failure

* It occurs when hard-disk drives or storage drives used to fail frequently. It was a common problem in the early day of technology evolution.
* It occurs due to the formation of bad sectors, disk head crash, and unreachability to the disk or any other failure, which destroy all or part of disk storage.

Log Based Recovery in DBMS

Log is nothing but a file which contains a sequence of records, each log record refers to a write operation. All the log records are recorded step by step in the log file. We can say, log files store the history of all updates activities.

Log contains start of transaction, transaction number, record number, old value, new value, end of transaction etc.

If within an ongoing transaction, the system crashes, then by using log files, we can return back to the previous state as if nothing has happened to the database.

The log is kept on disk so that it is not affected by failure except disk and failures.

Different types of log records are as following -

* $< T_i, X_j, V1, V2 >$

  T = Transaction
  X = Data
  V1 = old Value
  V2 = New Value

* $< T_i, \text{Start} >$ → T_i starts Execution
* $< T_i, \text{Commit} >$ → T_i is Committed
* $< T_i, \text{abort} >$ → T_i is aborted

Undo and Redo Operations

Because all database modifications must be preceded by creation of log record, the system has available both the old value prior to the modification of the data item and new value that is written for data item. This allows system to perform redo and undo operations as appropriate -

* Undo :- Using a log record sets the data item specified in log records to old value.
* Redo :- Using a log record sets the data item specified in log record to new value.

Log Based Recovery uses one of the techniques -

* Deferred Database Modification

It modifies the database after completion of transaction. The database modification is deferred or delayed until the last operation of the transaction is executed. Update log records maintain the new value of the data item

Recover system uses one operation which is as follows -

* Redo (Ti) :- All data items updated by the transaction Ti are set to a new value.

Immediate Database Modification

It modifies the database after a write operation, database modification is immediately done when a transaction performs an update/write operation. Update log records maintain both old and new values of data items.

The recovery system uses two operations, which are as follows -

* Undo (Ti) :- All data items updated by the transaction Ti, are set to old value.
* Redo (Ti) :- All data items updated by the transaction Ti, are set to a new value.

Shadow Paging

It is one of the technique that is used to recover from failure. It helps to maintain database consistency in case of failure.

Concept of Shadow Paging

Step 1:- Page is a segment of memory. Page table is an index of pages. Each table entry points to a page on the disk.

Step 2:- Two page tables are used during the life of a transaction, the current Page table and the shadow Page table. Shadow Page table is a copy of the current page table.

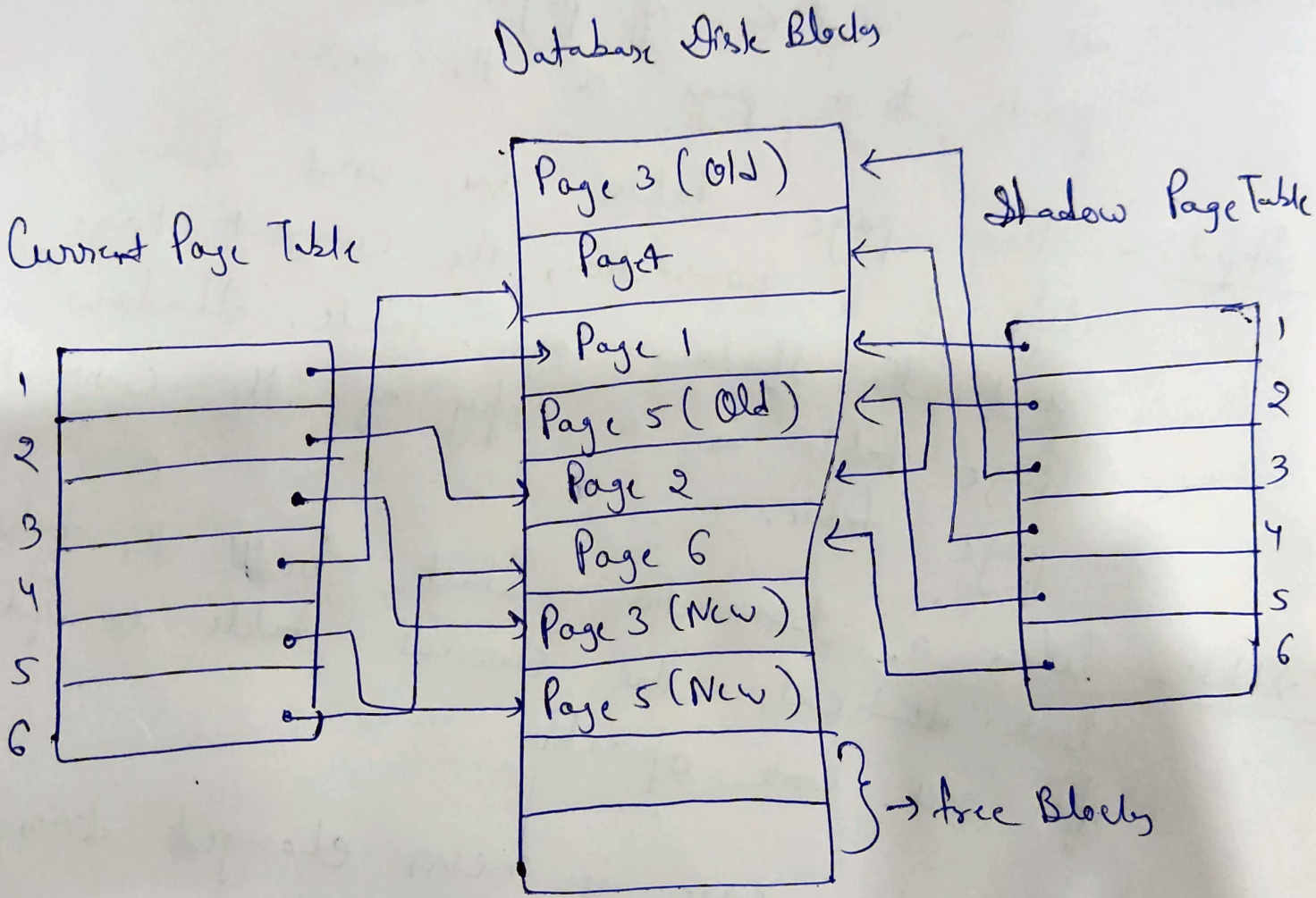
Step 3:- When a transaction starts, both the tables look identical, the current table is updated for each write operation.

Step 4:- The shadow page is never changed during the life of the transaction.

Step 5:- When the current transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released.

Step 6 :- The shadow Page table is stored in non-volatile memory. If the system crash occurs, then the shadow page table is copied to the current Page table.

The shadow paging is represented diagrammatically as follows :-



Advantages :-

* No need for log Records.
* No Undo Redo Algorithm
* Recovery is faster.

Disadvantages :-

* Data is fragmented or scattered.
* Garbage collection Problem. Database pages containing old versions of modified data need to be garbage collected after every transaction.
* Concurrent transaction are difficult to Execute.

Recovery With Concurrent Transactions

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 application state so that it can restart from that point in case of failure.
* Checkpoint is a point of time at which a record is written onto the database from the buffers.
* Checkpoint shortens the recovery process.
* When it reaches 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 the next checkpoint and so on.
* The checkpoint is used to declare the point before which the DBMS was in the consistent state, and all the transactions were committed.

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.