20/12/21

UNIT-4

Transaction :-

Transaction is a set of logical operations. It contains group of blocks. A Transaction is an action or series of actions. It performed by a single user to perform operations for accessing the database.

For example :-

An employee of bank transfer Rs. 800/- from X a/c to Y a/c

X account :-

open_account(X) =

old_balance = X balance

new_balance = old_balance - 800

X_balance = New_balance

close_account(X)

Y account :-

open_account(Y)

old_balance = X balance

new_balance = old_balance + 800

X_balance = new_balance

close_account(Y)

Operations of transactions:-

Read(x):- Read operation is used to read, data from database and stored in a buffer in main memory (Temporary m/m) DB → buffer in m/m

Write(x):- Write operation is used to write the value back to the database, from buffer.

buffer in m/m → DB.

```
    x = 1000;
m/m { 1. R(x);
    { 2. x = x - 500;
      W(x)
```

For example:- In the above transaction, The debit transaction fails after executing operation 2. x values will remain in the database which is not acceptable by the bank. To solve the problem we use two important transaction for operations:-

1. Commit:- It is used to save the work done (T2) permanently.
2. Rollback:- It is used to undo the work (T1) done. (shows the already existed data)

Storage structures

1. Non-volatile (floppy disk, hard disk)
2. Volatile
3. Failure classifications

volatile - temporary m/m, small amount of m/m.

(i) Transaction failures → logical errors
                        → system errors

(ii) System failures

(iii) Disk failures.

21/12/21

Transaction properties :-

A C I D

1. Atomicity
2. Consistency (without any interuptions)
3. Isolation
4. Durability

Atomicity :- All the operations of a transaction take place at once or not. If not the transaction the given process is abort.

→ There is no midway (partial execution) i.e..., the transaction cannot occur partially, each treated as one unit and either run to completion or it not executed all.

→ Atomicity involves two operations
(i) abort - if transaction aborts then all changes are not visible
(ii) commit

Consistency :-

→ The integrity constraints are maintained that the database is consistent before and after the transaction.

→ The execution of transaction will leave a database in either its prior (previous) stable state

or new stable state.

→ The transaction is used to transform the DB from one consistent state to another consistent state.

Inconsistency - T1 success & T2 fails
consistency - Both success.

Isolation:-

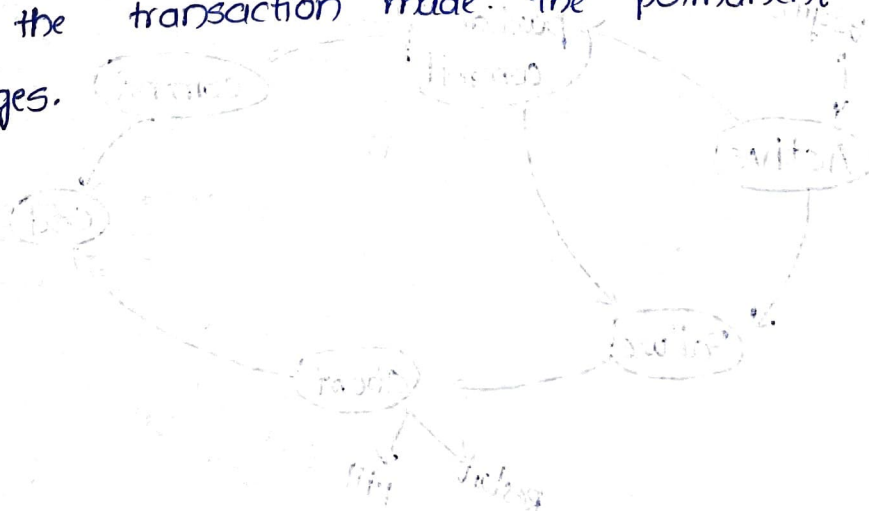
→ It shows that the data which is used at the time of execution of transaction cannot be used by the second transaction until 1st one is completed.

T1		T2
R(A)		R(B)
A = A - 100	⟶	A = A + 100
W(A)		W(B)

→ In isolation the transaction T1 is being executed then the data item cannot be accessed another transaction T2 until T1 ends.

Durability:-

The durability property is used to indicate the performance of DB in consistent state. It states that the transaction made the permanent changes.



22/12/21

Serialization:-

- Schedule (set of chronological order)
- serial schedule (one after another)

When multiple transactions are being executed by the OS, in a multi programming environment. This are possibilities that instructions of one transaction are interleaved with some other transaction.

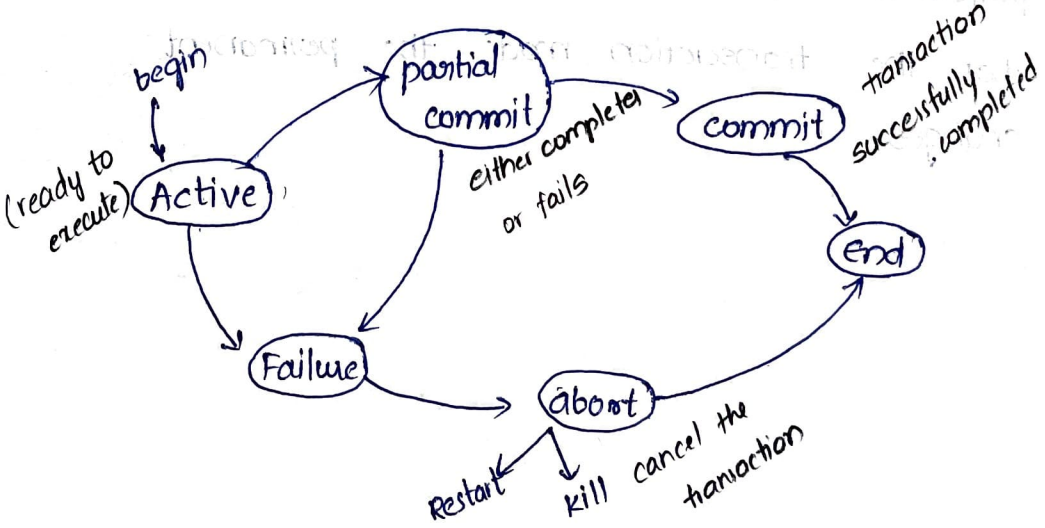
Schedule:- A chronological sequence of a transaction called schedule.

serial schedule:- It is a schedule in which transaction are alined in such a way that one transaction is executed first. when a first transaction completed its cycle then the next transaction is executed.

States of transaction:-

execution prgm completed but doesn't stored in DB

completed - stored in DB



Active state :-

In this state transaction is being ready to execute.

partial commit :-

In this state transaction executes final operation but data is not saved in database.

commit :-

A transaction is said to be in commited state if it executes all operations successfully & permanently saved on DB system.

Failure :-

If any of the partial commit to failure states it makes by the database recovery system fields. Then the transaction is said to be failed state.

Abort :-

If any of the checks fail and transaction has reached a fail state then DB recovery system will make sure that DB is in previous consistent state, and if not then it will abort or roll back the transaction to bring in the database into a consistent state.

→ After abort the transaction the database recovery module will select restart the transaction or kill the transaction.

23/12/21

Transaction isolation levels:-

The transaction isolation levels are defined by the given data is preserved or absence. Based on transaction isolation levels following some requirements like.

1. Dirty read
2. Non-repeatable read
3. phanthon read.

1. Dirty read:-

It occurs when a transaction reads the data that has not yet been committed

For example:-

T1 updates a row and its uncommitted, T2 reads the updated row, if transaction (T1) roll backs the change, T2 will have read data is considered never to have existed.

T_1	T_2
Read(x) uncommited	
	Read(x)
Roll back	
	committed

L1
L2
L3
L4

T_1	T_2
Read(x)	
x = x-500	
w(x) 2700	
uncommited	
	Read(x)
	x = x+500 3500
	w(x)
3000 Roll back	
	committed 8700

2. Non-repeatable read :-

A non-repeatable read occurs when a transaction reads same row wise and get different values each time. (simultaneous transaction)

Suppose T1 reads data due to concurrency another Transaction T2 updates the same data and commit. If Transaction T1 reads the same data it retrieves different values.

T1	T2
Read(x)	
	Read(x)
write(x)	
	read(x)

3. phanthom read :-

When two same queries are executed but row retrieve by the two or different

For ex :- Transaction T1 retreives a set of rows that satisfy same such criteria. Now T2 generates some new rows that match the search criteria for transaction 1. If Transaction T1 re-executes the statements reads the rows & will get different values in the given transaction.

T1	T2
Read(x)	
	Read(x)
Delete(x)	
	Read(x)

4 levels of Transaction isolation levels

1. Read uncommitted
2. Read committed
3. Repeatable read
4. serializable.

24/12/21

Read uncommit :-

It is the lowest isolation level and in the level one transcation may read not yet commited changes made by other transactions.

Read committed :-

Isolation level guarantees of data transaction and any data reads committed at the moment of read transaction.

Repeatable read :-

It is the most restrictive and standardizied isolation. The transaction holds read lock on all rows in preferences and write lock on all rows it insert, update and delete. since other transaction can't read, update or delete. those rows consequently to avoid non-repeatable rows

Serializable :-

This is the highest isolation level and a serilizable execution is guaranteed to be a serilizable.

→ The serializable execution is defined to be a execution of operations in which concurrently executing transaction appears to be serially executed.

Concurrency control:-

concurrency control means control the simultaneous execution in the given DB.

→ We have to define before concurrency control, we should know about concurrency execution.

→ In a multi-user system, multiple users can access and use same DB at one time.

Problems with concurrent transaction:-

1. Lost-update problems (W-W conflict):-

Ex:- consider two transactions (T_x, T_y) performs read and write operation an account of A

Time	T_x	T_y
T_1	Read(x) 300	
T_2	x = x-50	
T_3		Read(x)
T_4		x = x+50
T_5	250 write(x)	
T_6	lost	write(x) 350

The problem occurs when two different transactions performed read write operations on

some DB items. In the interleaved manner that makes the value of item incorrect. Hence making the DB is inconsistency.

27/12/21

2. Dirty-read problem:-(W-R conflict)

The dirty read problem occurs when one transaction updates on of the database, and sometimes the transaction fails and before the data gets roll back.

→ The updated DB item is accessed by another transaction

Time	T_1	T_2
T_1	R(A) -300	
T_2	300 A = A+50	
T_3	W(A)-350	
T_4		R(A) - 350
T_5	roll back (or) server down (300)	
T_6		commit -350

3. Unrepeatable read problem:-

Two different values are read for same database items.

Time	T_1	T_2
T_1	R(A)-300	
T_2		R(A)-350
T_3		A=A+50-350
T_4		W(A)
T_5	R(A)-350	

Concurrency control :-

concurrency control is the working concept required for the controlling and managing the concurrent execution of data base operation and avoid inconsistency of database and maintain concurrency of the database using concurrency control protocals.

→ concurrency control protocals ensured that atomicity, consistency, durability and isolation & seriliazability of the ~~currency~~ concurrency execution transactions

→ this protocals categorized by 3 types :-

1. lock base concurrency control protocal
2. Time stamp " " "
3. Validation based " "

29/12/21

Lock based concurrency control :-

→ Locking :- locking is a process is used to control the concurrent access to data where one transaction is accessed in database.

→ lock based protocal :- In the protocal can't read and write the data until it acquires an appropriate locks.

There are two types of locks :-

1. shared lock
2. Exclusive lock

1. shared lock :- lock S(A) - Read

shared lock is also known as read only lock. In a shared lock, the data item can only read by the transaction.

→ It can be shared b/w transactions because when the transaction holds a lock, then it cannot update the data on the data item.

→

		or		
slock	read(A)		lock	S(A) → shared
	R(A)			R(A)
unlock	read(A)			unlock (A)

2. Exclusive lock :- lock X(A) - read & write

In the EL the dataitem will both read & write by the transaction.

→ the lock is exclusively and in this lock multiple

transactions do not modify of the same data simultaneously.

lock read (A)

A = A+50

lock write (A)

unlock write (A)

→ There are 4-types of lock-based protocols available, In the lock-based, concurrency control.

1. simplistic lock based protocal :-

2. The simplistic lock protocal allows all the transactions to get the lock on the data before insert, update and delete and it will unlock the data item after completion of transactions

→ 2. conversion of locks :-

There are two types of locks :-

(i) upgrading locks

(ii) downgrading locks.

- The upgrading locks, transaction starts from shared locks to exclusive locks (or) read to write.
- Downgrading locks starts from exclusive locks to shared locks (or) write to read.

→ 3. Lock compatibility matrix :-

When the locks apply in simultaneous transactions only read to read / shared to share transactions are acceptable

	S	X
S	✓	X
X	X	X

3/9/22

Lock based protocals:-

2. Pre claiming lock protocal:-

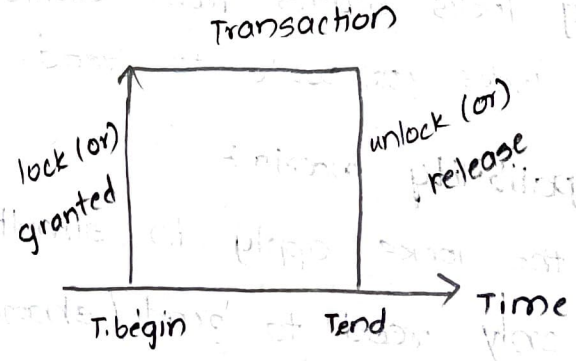
→ Pre claiming lock protocal evalute the transaction to list all the data items on which they need locks.

→ Before initiating, on execution of the transaction it requiest to database for all locks on those data items.

→ If all the locks are granted then this protocal allow transaction to begin

→ When the transaction is completed then it release all locks.

→ If all the locks are not granted then this protocal allows the transcation to roll back and wait unit all the locks are granted.



3. Two phase locking (2PL) :- → growing phase, → shrinking phase

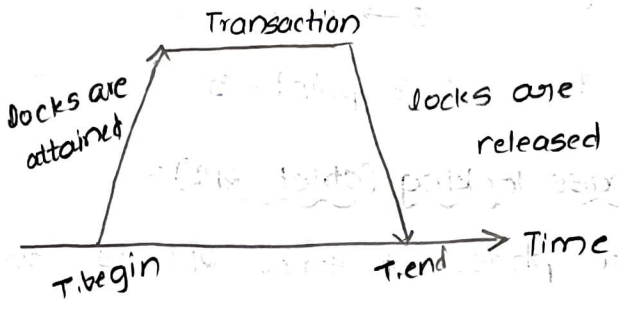
The 2PL protocal divides the execution phase of transaction into 3 parts.

i) When the execution of transaction starts it seeks permission for all locks its required.

ii) The transaction receives all locks, and

iii) It starts the 3rd phase is started as soon as transaction release its first lock.

→ iv) The transaction cannot demand any new locks, Its only release the received/acquired locks.



Growing phase :-

In the growing phase new locks on the data items may be acquired on the transaction but can't release the data items.

Shrinking phase :-

The existing locks held by the transaction may be released but no new locks cannot acquired (or) released.

Ex :- upgrading lock → S(A) to X(A) → growing phase

downgrading lock → X(A) to S(A) → shrinking phase

Time	T_1	T_2
1.	lock S(A)	
2.		lock S(A)
3.	lock X(B)	
4.	unlock (A)	
5.		lock-X(C)
6.	unlock (B)	
7.		unlock (A)
8.		unlock (C)

T_1 : T_2

$g \rightarrow 1-3$ $\quad$ $g \rightarrow 2-5$

$s \rightarrow 4-6$ $\quad$ $s \rightarrow 7-8$

lock point – 3 $\quad$ lock point - 5

4. Strict-two-phase locking (strict 2PL) :-

The first phase of strict 2PL is similar to two phase locking protocals. after

→ In first phase after receive all the locks the transaction to execute normally.

→ The difference b/w strict 2PL and 2PL does not release the lock after using it.

→ strict 2PL wait until the whole transaction to commit & then it release all the locks at a time.

→ strict 2PL protocal doesn't having shrinking phase of lock releases.

4/1/22

Time stamp ordered protocal:-

Time stamp is known as uniquely identify to created by DBMS in the transaction

→ The time stamp ordered protocal is used to order the transactions based upon their time stamps.

→ The order of transaction nothing but either ascending or descending order of the transaction creation.

For example:- There a two transactions T1 & T2 in that the Transaction T1 entered the system at 07 time and Transaction T2 entered the system at 09 time.

→ The T1 has highest priority so it's executes first as it is entered the system first

→ There are two time stamp ordering protocals.

1. T_i issues a Read (x) operation

Read & write <= $TS(T_i)$

- If $W\_TS(x) > TS(T_i)$ - reject
- If $W\_TS(x) <= TS(T_i)$ - accept

2. T_i issues a write (x) operation

- $TS(T_i) < R\_TS(x)$ - rejected
- $TS(T_i) < W\_TS(x)$ - reject

Validation based protocal:- after read before write

The validation based protocals the transaction is executed by read phase, validation phase and write phase

Read phase:- In this transaction T_i is read & executed and it is used to read the values of various data items and stored to temporary local variable.

validation phase:- The temporary variable values will be validate against the actual value.

Write phase:-

If the validation of transaction is validated, the temporary results are return to the databases.

5/1/22

Multiple Granularity:-

Multiple Granularity defines a hierarchally break-up the database into blocks and which can be a locks.

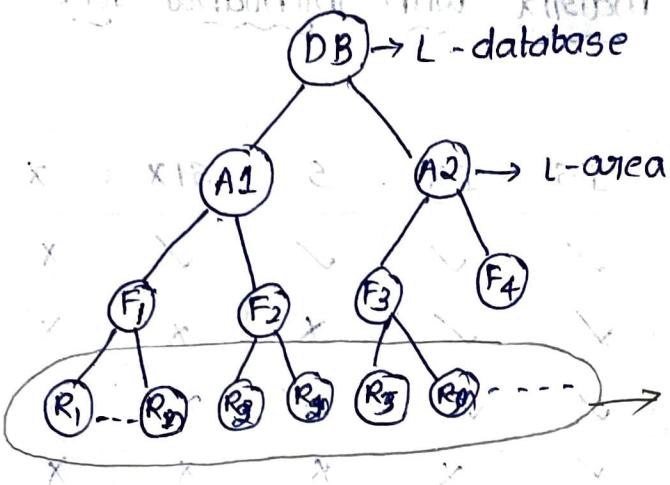
→ The multiple Granularity protocal enhanced concurrency control and reduce lock overheads.

→ It maintains what two locks and how two locks of tracking in transaction.

→ It make easy to decide either a lock a data item or unlock a dataitem.

→ In the multiple granularity, 4 levels of nodes

1. Database ↓
2. Area of DB ↓
3. Files ↓
4. Records ↓



6/1/22 There are 3 types of levels in multiple granularity :-

1. Intention shared lock (IS)
2. Intention exclusive lock (IX)
3. shared intention exclusive lock (SIX)

Intention shared mode :-

It contains explicit locking of the lower level in the tree and there is used only shared locks.

Intention exclusive :-

It contains explicit locking at the lower with exclusive or shared locks.

shared & intension exclusive :-

In this lock, the node is locked in shared mode and some nodes locked in exclusive in same transaction.

Compatability matrix with intentional lock mode:-

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
S	✓	X	✓	X	X
SIX	✓	X	X	X	X
X	X	X	X	X	X

→ observe that multiple granularity the locks are quiet acquired in top-down order and locks must release bottom to top order.

→ for example, If Transaction t1 reads records Ra1 in file Fa then transaction t1 needs to lock the database, A1 (area), file in intention exclusive mode. Finally it needs to lock Ra1 in S mode (shared mode)

→ Transaction t2, modified records Ra1 in the file Fa, then it can do after locking DB, area, file in IX mode, finally it needs the Ra1 in X mode. (exclusive mode)

→ transaction t3, needs all the records in file Fa then lock the DB, area, file in IS mode, atleast it needs to lock FA in shared mode.

→ The transaction t_4, reads the entire database then the T_4 needs to lock the DB in shared mode.

Assignment :-

What is deadlock?

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give uplocks. Deadlock is said to be one of the most ____ complications in DBMS as no task ever gets finished and is in waiting state forever.

Ex:- In student table transaction T_1 holds a lock on some rows and needs to update some rows in the grade table simultaneously transaction T_2 holds locks on some rows in the grade table and needs to update the rows in the student table held by Transaction T_1.

Now, the main problem arises. Now transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 to release its lock. All activities come at halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transaction.

2. What is deadlock avoidance?

When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restating 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 the method is suitable only for the smaller database. For larger database, deadlock prevention method can be used.

3. What is deadlock prevention?

Deadlock prevention method is suitable for a large database. If the resources are allocated in such a way that deadlock never occurs, then deadlock can be prevented.

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