MVI C10 Q. Fundamental properties of Transaction / ACID properties.

- To understand transaction properties, we consider a transaction of transferring 100 rupees.
- Let Ti be a transaction that transfers 100 from account A to account B. This transaction can be defined as,
  a). Read balance of account A.
  b) Withdraw 100 rupees from account A & write back result of balance update
  c) Read balance of account B
  d) Deposit 100 rps to account B & write back result of balance update.

- 4 transactions properties:
  1. Atomicity
  2. Consistency
  3. Isolation
  4. Durability.

1.
- Transaction must be treated as a single unit of opn.
- When a sequence of opn are performed, they are treated as single large operation.
- ex: a). withdrawing money from account.
- execution should either complete or not executed at all
- no partial execution of transaction allowed

DATE / /

2. Consistency

- Consistent state is a db state in which all valid data will be written in db.
- If a transaction violates some consistency rules, the whole transaction will be rolled back and the db will be restored to its previous consistent state
- On the other hand, if transaction executed successfully, db state will go from consistent state to another one
- DBMS should handle inconsistency & ensure db is clean at the end of each transaction.
- This means transaction will never leave dbs in half finished state. If it does, it is rolled back

3. Isolation

- Isolation property ensures that each transaction must remain unaware of other concurrently executing transactions.
- It keeps multiple transactions separated from each other until completion.
- Operations happening in a transaction are invisible to other transactions until the transaction commits or rollback.
- Transaction isolation is configurable in a variety of mode.
- Even though many transactions may execute concurrently in a system.

System must guarantee that, for every transaction (Ti) all other transactions has finished before it or other transactions started execution after transaction (Ti) finished.
- Each transaction is unaware of other transaction executing in the system simultaneously.

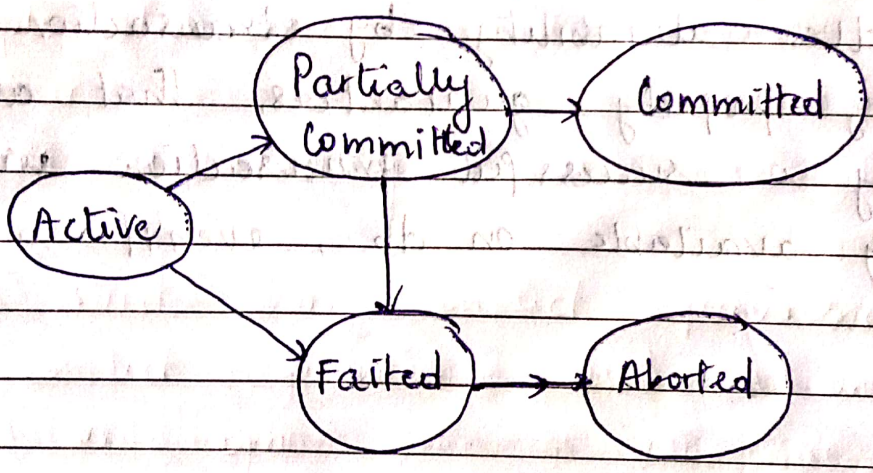
4. Durability.

- Results of transaction that has been commited to db will remain unchanged even after db fails.
- Those changes are permanent after commit end.
- DB handles durability by transaction log.
- Durability property guarantees, that all changes made by a successful transaction will be permanently available on db, even if a db failure occurs.

Q. Explain transaction state diagram.

- if transaction completed successfully → saved to db server ← called as committed transaction.
- committed transaction transfers db from one consistent state to other one.
- transaction may fail, such transaction is aborted
- if aborted transaction made changes, those must be undone or rolled back

States of transaction



Active:
- initial state
- transaction execution → active state
- trans^{n} remains in this state until it finishes.

Partially committed
- as soon as last opn in transaction is executed, transaction goes in partialy committed state.
- at this condition, transaction has completed its execution & ready to commit on db server. it can still be aborted.

Q. Explain the concept of seriability with its types.

1.

- The db system should have control above concurre execution of transactions, serializability will ensure that the db state remains in a consistent one.
- A schedule is the order in which the operations of multiple transactions appear for execution.
- Serial schedules are always consistent, non serial schedules are not always consistent.
- Seriakizability is a concept in db that helps identify which non serial schedules are correct & will maintain consistency of db.

- We consider only two opn for sake of computati simplicity: read & write.

- Seriakizability is mainly of two types:

  - Conflict serializability
  - View serializability.

Q. conflict serializability.

- A pair of consecutive db action (read / write) is in conflict if changing their order would change the result of at least one ~~last time~~ of the transactions
- Two operations are called as conflicting opn if all of the following conditions hold true for the
  - both the opn belong to different transact
  - both opn are on same data item
  - at least one of the operation is write opn.

- Example Technique:

Q. View serializability.

- less strict than conflict equivalence, but it is like conflict equivalence based on only the read and write opn.

Condition for view equivalence

Let D = data item
S_1, S_2 = Transaction schedules
T_i, T_j = Database transactions

Schedules S_1 S_2 are view equivalent if they satisfy the following condition for each data item D.

- S_1 & S_2 must have same transactions included, also they are performing same operations on same data. If T_i reads initial value of D in S_1, T_i also reads initial value of D in S_2

- If T_i reads value of D written by T_j in S_1, then T_i also reads value of D written by T_j in S_2

- If T_i writes final value of D in S_1, then T_i also writes final value of D in S_2.

- First 2 conditions ensure that transaction reads same value in both schedules
- Condition 3 ensure that final consistent state.

- If a concurrent schedule is view equivalent to a serial schedule of same transactions, then it is view serializable

- Give example

- Every conflict serializable schedule is view serializable but not vice versa

MU CII Q. Explain Two phase locking protocol & list its advantages

1) - In a concurrent environment, many users can use data in a DBMS simultaneously and each user feels he is having exclusive access to db.
- To achieve such system, there must be interaction among those concurrent transaction.
(1) Mutual exclusion
- Whenever one transaction is accessing data, second transaction should not change data. This can cause a 'dirty pb.
- This is why locking concept is applied here to assure that one process should not retrieve or update a record which another process is updating.

2) - Whenever any user transaction wants to access any data item (D) then he will issue a locking request to concurrency control manager
- A lock manager is a process that retrieves locking request message & sends response accordingly.
- Transaction can access data if it is locked by that transaction.
- Each transaction in the system should follow a set of rules called "locking protocol".
- It tells when every transaction should lock or unlock the data item.

Two phasing locking protocols

(1) - 2PL synchronizes read & write by explicitly detecting and preventing conflicts between ~~two~~ concurrent operations.
- Before reading data item X, transaction should 'own' a read lock on X. Before writing into X, transaction should 'own' a write lock on X.
- The ownership of locks is governed by 2 rules
  1). ≠ transactions cannot simultaneously own conflicting locks
  2) once a transaction surrenders ownership of lock, it may never obtain additional locks.
- Definition of conflicting locks depends on the type of synchronization being performed

- For RW, conflict when:
  - Both locks are on same data item
  - One is read lock and the other is write lock
- For WW, conflict when:
  - Both locks on same data
  - Both are WW locks.

(2) Two Phase locking protocol divides the execution phase of transaction into three parts
  i) when execution starts, it seeks permission for the lock it requires.
  ii) transaction acquires all the locks. 3rd phase is started as soon it releases its first lock.
  iii) Transaction cannot demand any lock, it only ~~finishe~~ releases the acquired locks.

- Two phases of 2PL

- Growing : a new lock on the data item may be acquired by the transaction but none can be released.
- Shrinking phase : existing lock held by transaction but no new locks can be acquired

Initially transaction is in growing phase, it acquires locks as needed.
The pt at which transaction obtain final (last) lock is called as lock pt of transaction.
⇒ end of growing phase & start of shrinking phase
- When transaction terminates, all remaining locks are automatically released. Some systems also require that transactions holds all locks until termination

(3) Advantages:
- ensures conflict serializability
- simple to implement & understand

(4) -Drawbacks:
- Deadlock may occurs in two phase locked schedule
- Cascaded roll back may occur under two phase locking
ex: if one transaction is rolling back, all transaction depending on that one will be rolled back

Q. Explain Timestamp Based protocol.

(1) - To achieve serializability, order of transaction for execution can be decided in advance using its time at which transaction entered in the system
- a general method to achieve this is using time stamp ordering protocol.

(2)

- A fixed timestamp is assigned at start of execution of transaction. Every transaction Tj has been assigned a timestamp by dbs denoted as Ts (Tj).
- A timestamp which has entered in system recently will have greater time stamp.
- System clock is used as timestamp or a logical counter can be used as timestamp and incremented after every assignment.
- Every data item x is with 2 timestamp values:
  a) W- timestamp:
  - it denotes the largest TS of any transaction that executed WRITE(x) successfully on given data item.
  - TS of recent write operation.
  b) R - TS
  - it denotes the largest TS of any transaction that executed READ(x) successfully on data item (x)
  - TS of recent read operation.

(4) Advantages:

- ensures serializability, as conflicting operations are processed in order of Tstamp ~~control~~ of opn
- ensures that it is free from deadlock

(5) Disadvantages:

- starvation is possible for long Transaction if short transaction conflicts with it, it causes restorative of long ~~opn~~ transaction again & again.
- it may not give recoverable schedules.

VI C12 Q Log based recovery

(1) - There can be problem in accessing database due to any reason.
- The most widely used structure for recording db modifications is Transaction log. (or log)
- The log is a sequence of log records, recording all the update activities done on the db by users.

(2) - Transaction log records are maintained to record various events during transaction processing.
- Transaction log records ~~are maintained~~ is recorded when transaction performs a write operation to record.
- log record to be useful for recovery from system and disk failures, the log must reside in stable storage.
- We assume that every log record is written to the end of opn on stable storage as soon log is created
- Transaction log contains following data:
  i) transaction identifier (T_i) unique identifier of transaction that is recorded at write operations.
  ii) Data Item Identifier(X): unique identifier to recognize data item written.
  iii) Old value (V_1): Value of old data item
  (iv) New Value (V_2): Value of new data item after write action

(3) - Types of log records:
  a) initial log record
  b) update log record
  c) completion log record
  d) Checkpoint record

a) Initial log record :
- this log indicates that recording log file is started
- log record : <Tn Start>, Transaction has started

eg: <T1 Start> : Transaction 1 has started

b) Update log record
- an update log record describes a single db write
- it also include the value of the bytes of pages before and after the page change.
- log record : <Tn, X, V1, V2>
- Transaction (Tn) has performed a write on data item X.
- It modify current value V1 of X to value V2.
eg: <T1, A, 100, 500> : Transaction T1 has changed value of A to 500

c) completion log (commit / abort log) record:
- if the transaction completes opn successfully, then commit
- if any problem while executing transaction decision to abort then roll back the transaction.
- Operational update log : <Tn commit> or <Tn abort>
- It commits or roll back the opn commited by transaction

d) Check point record
- records a pt when checkpt has been made
- these are used for speed up recovery.
- it also record information that eliminates the need to read a log's past
- log record : <Tn Checkpt A>
eg: <T1 Checkpoint A> : Transaction T1 is commited to server.

(4) Recovery actions

Redo operation:
- If a transaction has recorded commit operation before failure occurs, then transaction must be redone
- Recover action: Redo (T_i).
eg: REDO (T_i)

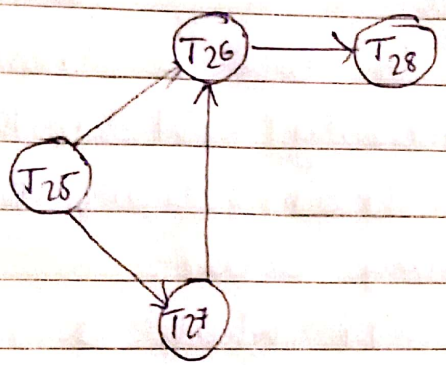
Undo operation:
- If a transaction has not recorded commit operation before failure occurs, then transaction must be undone
- Recovery Action: UNDO (T_i)
eg: UNDO (T_i).

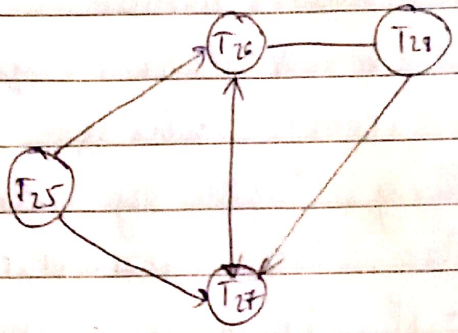
Q Deadlock Explain deadlock detection, prevention & recovery

(1) - In multiprocess system, ~~a system~~ deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.
- A system is said to be in state of deadlock if there exists a set of transaction such that every transaction in the set is waiting for another transaction to complete its execution.
- For example, assume a set of transactions {T0, T1, T2, T3}. T0 needs a resource X to complete its task. Resource X is held by T1, and T1 is waiting for resource Y, which is held by T2. T2 is waiting for resource Z which is held by T0. Thus, all the processes wait for each other to release resources.
- In case a system is stuck into deadlock, the transactions involved in the deadlock are either rolled back or restarted.

(2) Deadlock Detection
- A simple way to detect a state of deadlock is with the help of some deadlock detection algorithms.
- Deadlock can be detected using directed graph as wait-for-graph.
- One node is created in wait graph for each transaction that is currently executing.
- Whenever a transaction Ti is waiting to lock an item X that is currently locked by a transaction Tj, a directed edge (Ti → Tj) is created.
- When Tj releases the lock(s) on items that Ti was waiting for, the directed edge is dropped.

- We have a state of deadlock if & only if the wait for graph has a cycle. Then each transaction involved in the cycle is said to be deadlocked
- To detect deadlocks, the system needs to maintain the wait-for graph and periodically to invoke an algorithm that searches for a cycle in graph
- To illustrate these concepts, consider the wait for graph in figure





- T_{25} waiting for T_{26}, T_{27}
- T_{27} waiting for T_{26}
- T_{26} waiting for T_{28}

No cycle, no deadlock state

Suppose now T_{28} is requesting an item held by T_{27}
Edge T_{28} - T_{27} added
Graph contains cycle hence deadlock

- The invoking of deadlock detection algorithm depends on two factors.
  - occurence of deadlock
  - numbers of transaction affected by it.

3) Deadlock prevention.

- To prevent deadlock situation in the system, the DBMS aggressively inspects all the opn where transactions are about to execute.

- DBMS inspects the operations and analyses if they can create a deadlock situation. If it finds that a deadlock may occur, then the transaction is never allowed to be executed.
- There are deadlock prevention schemes that use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation.

- Wait - die - scheme:
- if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one one the two possibilities may occur:
  - if $T_i < T_j$ - that is T_i which is requesting a conflicting lock, is older than T_j - then T_i is allowed to wait until the data-item is available.
  - if $T_i > T_j$ - that is T_i younger Then T_j - then T_i dies. T_i is restarted later with a random delay but with the same time stamp. This scheme allows the older transaction to wait but kills the younger ones.

- Wound - wait - scheme:
- if a transaction requests to lock a resource (data item) which is already held with one of conflicting lock by some another transaction, this may occur

- If $T_i < T_j$, then T_i forces T_j to be rolled back - that is T_i wounds T_j. T_j is restarted later with random delay but with same timestamp
- If $T_i < T_j$, then T_i is forced to wait until the resource is available.

This scheme allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item. In both cases the transaction that enters the system at a later stage is aborted

Deadlock Recovery.

a). Termination of processes.
-if deadlock is detected then a transaction or more are selected as victims for termination for the cycle of process
- Transaction with minimum costs are selected for rollback
- cost can be detected by following factors
  - for how much time transaction has computed
  - for how many data items it has locked
  - how many data item it may need ahead
  - number of transaction to be rolled back

b). Rollback
Once victims are decided, two ways to proceed
i) total rollback : Transaction is aborted and restarted
ii) partial rollback: rollback only transaction which is needed to break deadlock.

This approach needs to record some info like state of running transaction, locks on data item held by them. & deadlock detection algorithm specifies points up to which transaction to be rollback and recovery method has to rollback

e) Starvation

- it may happen that every time same transaction is selected as victim and it may lead to starvation of that transaction.
- system should take care that every time same transaction should not be selected as victim. So it will not starved.