Internal Assesment Test – III, December 2020									
Sub:	DATABASE MANAGEMENT SYSTEM							Code:	18MCA31
Date:	11-12-2020	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA

Answer Any 5 QUESTIONS		Marks	OBE	
			CO	RBT
1.	Define: Transaction. Explain ACID properties of transaction.	10	CO4	L1
2.	With the help of state transition diagram, explain the states of transaction execution.	10	CO4	L1
3.	Explain briefly about all lock-based protocols.	10	CO4	L1
4.	Discuss briefly about failure classification and recovery algorithm.	10	CO4	L4
5.	Explain in detail about database modification.	10	CO4	L4
6.	Explain how to deal with deadlock in concurrent control mechanism.	10	CO4	L1
7.	What are the levels of transaction isolation, do you have? How do you implement isolation levels? Explain.	10	CO4	L2
8.	Explain about recovery and atomicity.	10	CO4	L1

CMR Institute of Technology, Bengalore – 560 037
Department of Computer Applications
Scheme and Solutions for Internal Assessment Test – III

18MCA31 – DATABASE MANAGEMENT SYSTEM **Semester / Section: III** **Date of Test: 11-12-2020**

---

1. Define: Transaction. Explain ACID properties of transaction. (10)

- Collections of operations that form a single logical unit of work
- a unit of program execution that accesses and possibly updates various data items
- ACID properties of the transactions:

i. Atomicity: Either all operations of the transaction are reflected properly in the database, or none are. ("All-or-Nothing property")

ii. Consistency: Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database.

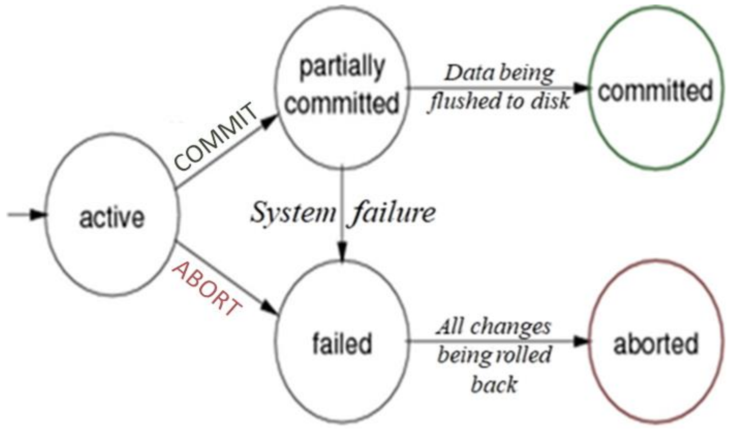
iii. Isolation: Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj, it appears to Ti that either Tj finished execution before Ti started or Tj started execution after Ti finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.

iv. Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures

Scheme:
Definition of transaction: 2 marks
ACID properties: 4 x 2 = 8 Marks

2. With the help of state transition diagram, explain the states of transaction execution. (10)

- a transaction may not always complete its execution successfully. Such a transaction is termed aborted
- Once the changes caused by an aborted transaction have been undone, we say that the transaction has been rolled back
- A transaction that completes its execution successfully is said to be committed



- A transaction must be in one of the following states:
  - Active: the initial state; the transaction stays in this state while it is executing.
  - Partially committed: after the final statement has been executed.
  - Failed: after the discovery that normal execution can no longer proceed.

- Aborted: after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- Committed: after successful completion

Scheme: Explanation: 8 marks and diagram: 2 Marks

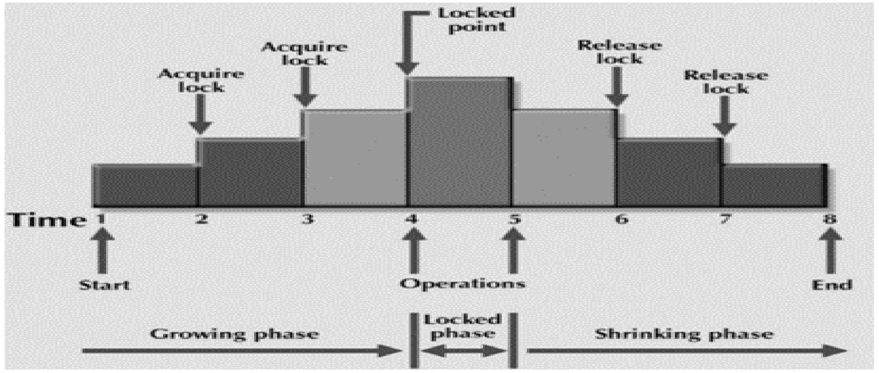
3. Explain briefly about all lock-based protocols. (10)

- Lock-Based Protocols: One way to ensure isolation is to require that data items be accessed in a mutually exclusive manner - while one transaction is accessing a data item, no other transaction can modify that data item

(i) The Two-Phase Locking Protocol: ensures serializability

- requires that each transaction issue lock and unlock requests in two phases:

1. Growing phase: A transaction may obtain locks, but may not release any lock.
2. Shrinking phase: A transaction may release locks, but may not obtain any new locks



- the two-phase locking protocol ensures conflict serializability
- Lock point: The point in the schedule where the transaction has obtained its final lock
- transactions can be ordered according to their lock points - a serializability ordering for the transactions
- Two-phase locking does not ensure freedom from deadlock

(ii) Strict two-phase locking protocol: a modification of two phase locking to avoid Cascading rollbacks

- It requires that:
- locking be two phase
- all exclusive-mode locks taken by a transaction be held until that transaction commits
- rigorous two-phase locking protocol: Another variant of two-phase locking which requires that all locks be held until the transaction commits

Scheme: each protocol carries 5 marks (2 x 5 = 10 Marks)

4. Discuss briefly about failure classification and recovery algorithm. (10)

- Failure Classification:
- There are various types of failure that may occur in a system, each of which needs to be dealt with in a different manner.
- the following are the types of failure:

(i) Transaction Failure: There are two types of errors that may cause a transaction to fail:

- Logical error: The transaction can no longer continue with its normal execution because of some internal condition, such as bad input, data not found, overflow, or resource limit exceeded.
- System error: The system has entered an undesirable state (for example, deadlock), as a result of which a transaction cannot continue with its normal execution. The transaction, however, can be re-executed at a later time

(ii) System crash: hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt (called as fail-stop assumption)

(iii) Disk failure: A disk block loses its content as a result of either a head crash or failure during a data-transfer operation.

- Copies of the data on other disks, or archival backups on tertiary media, such as DVD or tapes, are used to recover from the failure

- Determine the recovery from failure:
  - identify the failure modes of those devices used for storing data
  - consider how these failure modes affect the contents of the database
  - propose algorithms (recovery algorithms) to ensure database consistency and transaction atomicity despite failures
  - recovery algorithms, have two parts:

i. Actions taken during normal transaction processing to ensure that enough information exists to allow recovery from failures.

ii. Actions taken after a failure to recover the database contents to a state that ensures database consistency, transaction atomicity, and durability.

Scheme: failure classification: 5 Marks and explanation: 5 Marks

5. Explain in detail about database modification. (10)

- Database Modification
- Log provides recovery mechanism by means of the following steps in modifying a data item:

i. The transaction performs some computations in its own private part of main memory.

ii. The transaction modifies the data block in the disk buffer in main memory holding the data item.

iii. The database system executes the output operation that writes the data block to disk.

- a transaction modifies the database if it performs an update on a disk buffer, or on the disk itself;
- updates to the private part of main memory do not count as database modifications.
  - If a transaction does not modify the database until it has committed, it is said to use the deferred-modification technique.
  - If database modifications occur while the transaction is still active, the transaction is said to use the immediate-modification technique
- Deferred modification has the overhead that transactions need to make local copies of all updated data items;
- if a transaction reads a data item that it has updated, it must read the value from its local copy.
- The recovery algorithms support immediate modification.

- A recovery algorithm must take into account a variety of factors, including:
  - The possibility that a transaction may have committed although some of its database modifications exist only in the disk buffer in main memory and not in the database on disk.
  - The possibility that a transaction may have modified the database while in the active state and, as a result of a subsequent failure, may need to abort.
- the system to perform undo and redo operations as appropriate, based on the values of log record:
  - Undo: using a log record sets the data item specified in the log record to the old value
  - Redo: using a log record sets the data item specified in the log record to the new value

Scheme: Explanation: 10 Marks

6. **Explain how to deal with deadlock in concurrent control mechanism.** **(10)**

- A system is in a deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set
- There are two principal methods for dealing with the deadlock problem:

(i) deadlock prevention protocol to ensure that the system will never enter a deadlock state

(ii) try to recover from deadlock state by using a deadlock detection and deadlock recovery scheme

- ▪ Deadlock Prevention:
- There are two approaches to deadlock prevention

- ensures that no cyclic waits can occur by ordering the requests for locks, or requiring all locks to be acquired together (called as no circular wait)
- performs transaction rollback instead of waiting for a lock
- Another approach: impose an ordering of all data items, and to require that a transaction lock data items only in a sequence consistent with the ordering
- A variation of this approach is to use a total order of data items
- use pre-emption and transaction rollbacks
- In pre-emption, when a transaction T_j requests a lock that transaction T_i holds, the lock granted to T_i may be pre-empted by rolling back of T_i, and granting of the lock to T_j
- To control the pre-emption, we assign a unique timestamp
- The system uses these timestamps only to decide whether a transaction should wait or roll back
- Two different deadlock-prevention schemes using timestamps have been proposed:

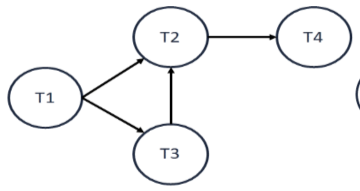
(i) The wait–die scheme: is a non-preemptive technique

- When transaction T_i requests a data item currently held by T_j, T_i is allowed to wait only if it has a timestamp smaller than that of T_j (i.e T_i is older than T_j). Otherwise, T_i is rolled back (dies).

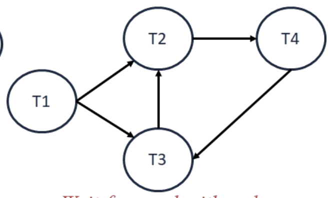
(ii) The wound–wait scheme is a pre-emptive technique

- When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp larger than that of T_j (that is, Ti is younger than Tj). Otherwise, Tj is rolled back (Tj is wounded by Ti).

- Deadlock detection and recovery
  - Deadlocks can be described precisely in terms of a directed graph called a wait-for graph
  - When transaction T_i requests a data item currently being held by transaction T_j, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph.
  - This edge is removed only when transaction T_j is no longer holding a data item needed by transaction T_i.
  - A deadlock exists in the system if and only if the wait-for graph consists a cycle



Wait-for graph with no cycle
(No deadlock is detected)



Wait-for graph with cycle
(Deadlock is detected)

- Recovery from Deadlock
- When a detection algorithm determines that a deadlock exists, the system must recover from the deadlock.
- The most common solution is to rollback one or more transactions to break the deadlock.
- Three actions need to be taken:

(i) Selection of victim

- determine which transaction (or transactions) to roll back to break the deadlock

(ii) Rollback

- Abort the transaction and then restart it
- Such partial rollback requires the system to maintain additional information about the state of all the running transactions

(iii) Starvation

- the same transaction is always picked as a victim.
- As a result, this transaction never completes its designated task (thereby creating starvation)

Scheme: Deadlock prevention, detection and recovery: 9 Marks
Diagram: 1 Mark

7. What are the levels of transaction isolation, do you have? How do you implement isolation levels? Explain. (10)

- The isolation levels specified by the SQL standard (SQL-92) are as follows:

i. Serializable: usually ensures serializable execution (default)

ii. Repeatable read: allows only committed data to be read and further requires that, between two reads of a data item by a transaction, no other transaction is allowed to update it. However, the transaction may not be serializable with respect to other transactions.

iii. Read committed: allows only committed data to be read, but does not require repeatable reads

iv. Read uncommitted: allows uncommitted data to be read. It is the lowest isolation level allowed by SQL

Implementation of isolation levels:

(i) Locking: Instead of locking the entire database, a transaction could lock only those data items that it accesses.

- Under such a policy, the transaction must hold locks long enough to ensure serializability, but for a period short enough not to harm performance excessively
- two kinds of locks: shared and exclusive
  - Shared locks: used for data that the transaction reads (read operation) and
  - exclusive locks: used for those it writes (write operation)

(ii) Timestamps: used to ensure that transactions access each data item in order of the transactions' timestamps if their accesses conflict

- For each data item, the system keeps two timestamps
  - The read timestamp of a data item holds the largest (that is, the most recent) timestamp of those transactions that read the data item.
  - The write timestamp of a data item holds the timestamp of the transaction that wrote the current value of the data item

(iii) Multiple Versions and Snapshot Isolation

- Multiple versions concurrency control (MVCC) keeps multiple copies of each data item

- It guarantees that all reads made in a transaction will see a consistent snapshot of the database
- the transaction itself will successfully commit only if no updates it has made conflict with any concurrent updates made since that snapshot

Scheme: Isolation levels: 5 Marks and Implementation: 5 Marks

8. Explain about recovery and atomicity. (10)

- perform either all or no database modifications made by T_i.
- However, if T_i performed multiple database modifications, several output operations may be required, and a failure may occur after some of these modifications have been made, but before all of them are made.
- To ensure atomicity: output to stable storage information describing the modifications, without modifying the database itself
- this information can help us ensure that:
  - all modifications performed by committed transactions are reflected in the database
  - no modifications made by an aborted transaction persist in the database.
- Log Records
- Log: The most widely used structure for recording database modifications
- The log is a sequence of log records, recording all the update activities in the database
- An update log record describes a single database write.
- It has these fields:
  - Transaction identifier: which is the unique identifier of the transaction that performed the write operation
  - Data-item identifier: which is the unique identifier of the data item written
  - Old value: which is the value of the data item prior to the write
  - New value: which is the value that the data item will have after the write
- Update log record is represented as: $<T_i, X_j, V_1, V_2>$ indicating that transaction Ti has performed a write on data item X_j which had value V_1 before the write, and has value V_2 after the write
- The other special log records exist of record significant events during transaction processing:
  - $<T_i$ start>: Transaction T_i has started
  - $<T_i$ commit>: Transaction T_i has committed
  - $<T_i$ abort>: Transaction T_i has aborted
- Whenever a transaction performs a write, it is essential that the log record for that write be created and added to the log, before the database is modified.
- Once a log record exists, we can output the modification to the database if that is desirable.

- Checkpoints
- When a system crash occurs, we must consult the log to determine those transactions that need to be redone and those that need to be undone
- need to search the entire log to determine this information
- two major difficulties with this approach:

(i) time-consuming process

(ii) we might unnecessarily redo transactions which have already output their updates to the database.

- Checkpoints are used to reduce these types of overheads
- A checkpoint is performed as follows:

1. Output onto stable storage all log records currently residing in main memory.
2. Output to the disk all modified buffer blocks.
3. Output onto stable storage a log record of the form <checkpoint L>, where L is a list of transactions active at the time of the checkpoint.

- The requirement that transactions must not perform any updates to buffer blocks or to the log during checkpointing can be bothersome, since transaction processing has to halt while a checkpoint is in progress
- A fuzzy checkpoint: is a checkpoint where transactions are allowed to perform updates even while buffer blocks are being written out.

Scheme: Log records: 5 marks / checkpoints: 5 marks