UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE &ENGINNERING

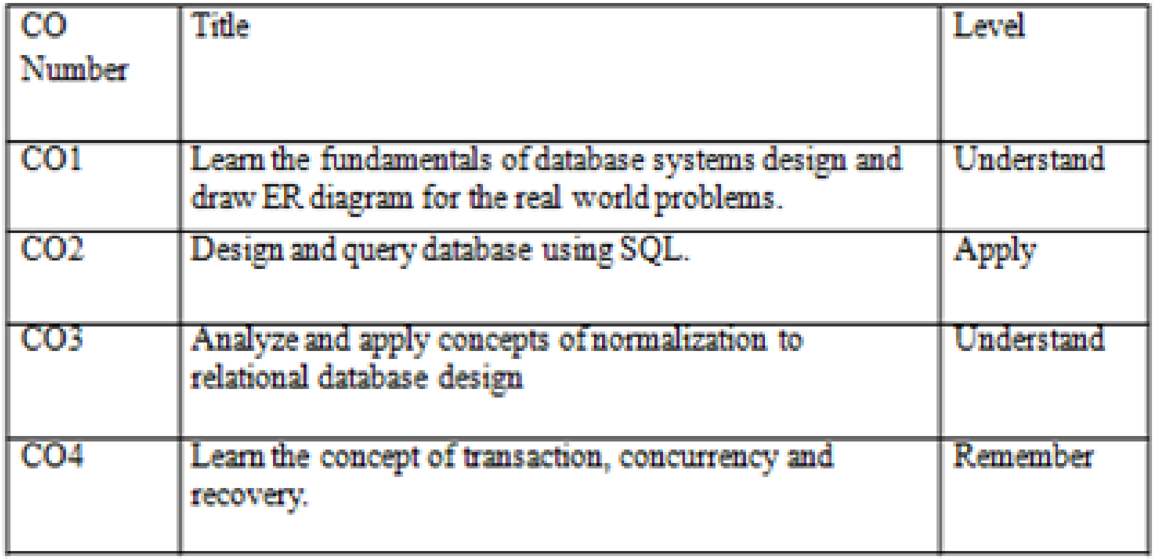
Bachelor of Engineering (Computer Science & Engineering)

Database Management System and CST-227

Prepared By: Ms. Heena Arora

Database Management System

Course Outcome



CO Number	Title	Level
CO1	Learn the fundamentals of database systems design and draw ER diagram for the real world problems.	Understand
CO2	Design and query database using SQL.	Apply
CO3	Analyze and apply concepts of normalization to relational database design	Understand
CO4	Learn the concept of transaction, concurrency and recovery.	Remember

Transaction Management and Concurrency Control

Introduction to Transaction Processing

Properties of Transactions

Serializability

Need for Concurrency Control

Locking Techniques

Time Stamping Methods

TRANSACTION

Transaction:

A transaction is an event which occurs on the database. Generally a transaction reads a value from the database or writes a value to the database.

For example, you are transferring money from your bank account to your friend’s account, the set of operations would be like this:

Simple Transaction Example

1. Read your account balance
2. Deduct the amount from your balance
3. Write the remaining balance to your account
4. Read your friend’s account balance
5. Add the amount to his account balance
6. Write the new updated balance to his account

PROPERTIES

ACID Properties:

- For each series of events involved in transactions have to follow the four basic properties called ACID properties. These are as follows:
  - **A means Atomicity**
  - **C means Consistency**
  - **I means Isolation**
  - **D means Durability**

ACID Properties

- Every event of transaction should follow these four properties called as ACID properties.
- A- Atomicity: That means either all transaction should take place in system or not any event of the transaction would be happened.

Cont.

Consistency- This means before and after the transaction consistency should be maintained.

Isolation- This means each transaction perform individually no other transaction disturbs its working.

Durability- This means if the system gets crash, the data item remain consistent with the help of using logs.

SERIALIZABILITY

When multiple events of transaction happened together and they want to use the same data item they must follow the principle of serializability. There should not exist any confliction.

If the two events wants to perform the read operation on same item than order is not so important.

If the two events wants to perform the write operation on same item than order is very important.

Transaction States

Transaction States:

There are the following six states in which a transaction may exist:

Active: The initial state when the transaction has just started execution.

Partially Committed: At any given point of time if the transaction is executing properly, then it is going towards it COMMIT POINT. The values generated during the execution are all stored in volatile storage.

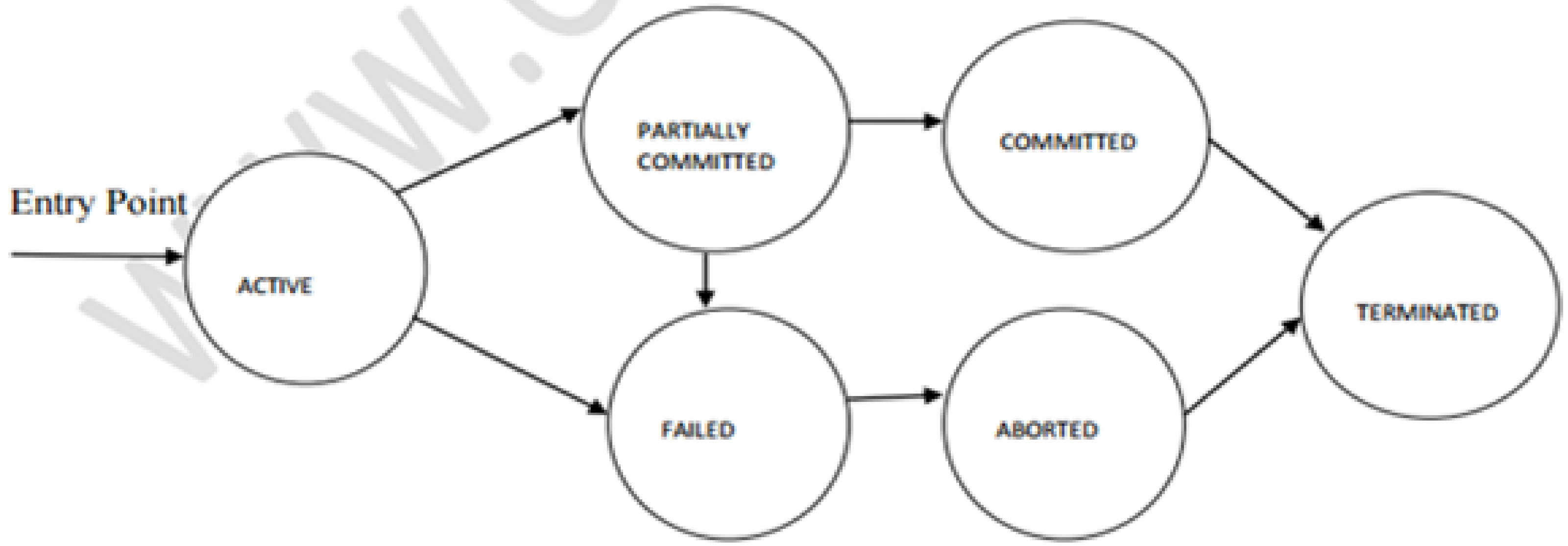
Failed: If the transaction fails for some reason. The temporary values are no longer required, and the transaction is set to **ROLLBACK**. It means that any change made to the database by this transaction up to the point of the failure must be undone.

CONT..

Transaction States:

- **Aborted:** When the ROLLBACK operation is over, the database reaches the BFIM. The transaction is now said to have been aborted.
- **Committed:** If no failure occurs then the transaction reaches the COMMIT POINT. All the temporary values are written to the stable storage and the transaction is said to have been committed.
- **Terminated:** Either committed or aborted, the transaction finally reaches this state.

CONT..



- Figure-1:Transaction States

Concurrency Control Algorithms

Concurrency Control Algorithms:

Locking

A Transaction “locks” a database object to prevent another object from modifying the object

Time-Stamping

Assign a global unique time stamp to each transaction

Optimistic

Assumption that most database operations do not conflict

Locking

- Lock is assigned to each and every data item of transaction.
- It is the responsibility of lock manager to assign the lock on data item.
- Lock is variable which is assigned to each data item. In Binary locking its has two states lock and unlock.

Types of Locks

Types of Locks: Binary:

- Binary Locks – Lock with 2 States
  - Locked – No other transaction can use that object
  - Unlocked – Any transaction can lock and use object

  All Transactions require a Lock and Unlock Operation for Each Object Accessed (Handled by DBMS)
  - Eliminates Lost Updates
  - Too Restrictive to Yield Optimal Concurrency Conditions

Problems of locking

Cause deadlocks

Transaction may be not serializable.

Problem can be solved by using algorithm of Two –Phase Locking Protocol.

Two Phase Locking

Two Phase Locking:

Defines how transactions Acquire and Relinquish Locks

1. Growing Phase – The transaction acquires all locks (doesn't unlock any data)
2. Shrinking Phase – The transaction releases locks (doesn't lock any additional data)

- Transactions acquire all locks it needs until it reaches locked point
- When locked, data is modified and locks are released

Deadlocks

Deadlocks:

- Occur when 2 transactions exist in the following mode:

  T1 = access data item X and Y

  T2 = Access data items Y and X

If T1 does not unlock Y, T2 cannot begin

If T2 does not unlock X, T1 cannot continue

T1 & T2 wait indefinitely for each other to unlock data

- Deadlocks are only possible if a transactions wants an Exclusive Lock (No Deadlocks on Shared Locks)

Controlling Deadlocks

Controlling Deadlocks:

- **Prevention** – A transaction requesting a new lock is aborted if there is the possibility of a deadlock – Transaction is rolled back, Locks are released, Transaction is rescheduled
- **Detection** – Periodically test the database for deadlocks. If a deadlock is found, abort / rollback one of the transactions
- **Avoidance** – Requires a transaction to obtain all locks needed before it can execute – requires locks to be obtained in succession

Classification of Locks:

Types of Locks: Shared / Exclusive Locks:

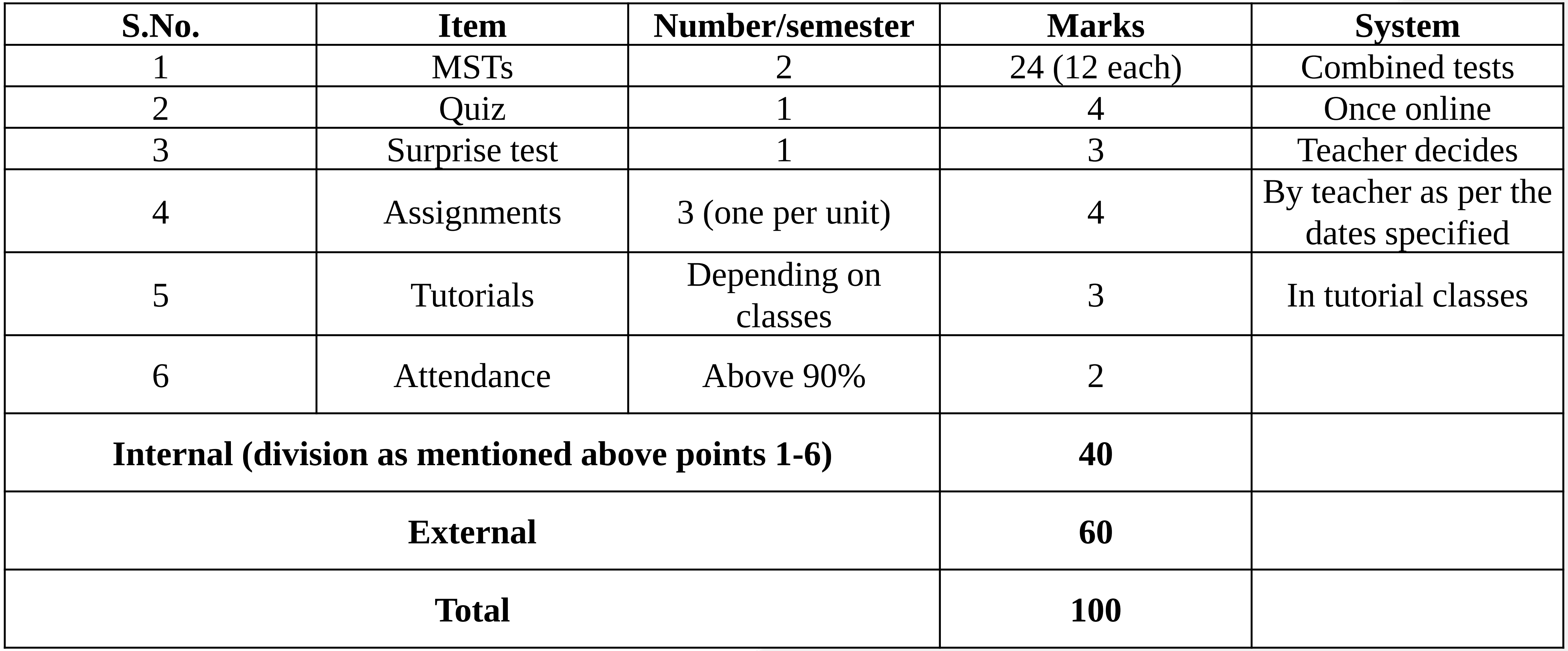
- Indicates the Nature of the Lock
- Shared Lock – Concurrent Transactions are granted READ access on the basis of a common lock.
- Exclusive Lock – Access is reserved for the transaction that locked the object.
- 3 States: Unlocked, Shared (Read), Exclusive (Write)
- More Efficient Data Access Solution
- More Overhead for Lock Manager
    - Type of lock needed must be known
    - 3 Operations:
        - Read_Lock – Check to see the type of lock
        - Write_Lock – Issue a Lock
        - Unlock – Release a Lock
    - Allow Upgrading / Downgrading of Locks

TIMESTAMP-BASED PROTOCOL

Timestamp-based Protocols

- The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.
- Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

Assessment Pattern



S.No.	Item	Number/semester	Marks	System
1	MSTs	2	24 (12 each)	Combined tests
2	Quiz	1	4	Once online
3	Surprise test	1	3	Teacher decides
4	Assignments	3 (one per unit)	4	By teacher as per the dates specified
5	Tutorials	Depending on classes	3	In tutorial classes
6	Attendance	Above 90%	2	
Internal (division as mentioned above points 1-6)			**40**	
External			**60**	
Total			**100**	

REFERENCES

- “Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles” by Narasimha Karumanchi
- “Data Structures Using C” by Aaron M. Tenenbaum
- “Data Structures and Algorithms” by Alfred V. Aho
- “Fundamentals of Data Structures in C by Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed

THANK YOU

For queries

Email: Heenae7725@cumail.in