Transaction Concepts – ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – Locking Protocols – Two Phase Locking – Deadlock – Transaction Recovery - Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery

LOCKING PROTOCOLS

- A lock is a variable associated with a data item that describe the statues of the item with respect to possible operations that can be applied to it.

- Locking is an operation which secures
  (a)Permission to Read
  (b) permission to Write a data item for a transaction.

- Example: Lock (X). Data item X is locked in behalf of the requesting transaction.

Unlocking is an operation which removes these permissions from the data item.

Example: Unlock (X): Data item X is made available to all other transactions.

- Lock and Unlock are Atomic operations.
- **Conflict matrix**

	Read	Write
Read	Y	N
Write	N	N

LOCKING PROTOCOLS

- **Lock Manager:**
  - Managing locks on data items.
- **Lock table:**
  - Lock manager uses it to store the identify of transaction locking a data item, the data item, lock mode and pointer to the next data item locked. One simple way to implement a lock table is through linked list
- **Types of lock**
  - Binary lock
  - Read/write(shared / Exclusive) lock
- **Binary lock**
  - It can have two states (or) values 0 and 1. 0 – unlocked
    - 1 - locked
  - Lock value is 0 then the data item can accessed when requested.
  - When the lock value is 1,the item cannot be accessed when requested.
- Lock_item(x)
- B : if lock(x) = 0 (* item is unlocked *) then lock(x) 1
- else begin
- wait (until lock(x) = 0) goto B;
- end;
- Unlock_item(x)
- B : if lock(x)=1 (* item is locked *) then lock(x) 0
- else
- printf (_already is unlocked _) goto B;
- end;
- **Read / write(shared/exclusive) lock**

LOCKING PROTOCOLS

- Read_lock
  - its also called shared-mode lock
  - If a transaction T_i has obtain a shared-mode lock on item X, then T_i can read, but cannot write ,X.
  - Outer transactions are also allowed to read the data item but cannot write.

- **Read_lock(x)**
- B : if lock(x) = ―unlocked‖ then
  - begin
  - lock(x) ―read_locked‖
    no_of_read(x) 1
  - else if
  - lock(x) =
    ―read_locked‖ then
  - no_of_read(x) no_of_read(x) +1
  - else begin
  - wait (until lock(x) =
    ―unlocked‖ goto B;
  - end;

LOCKING PROTOCOLS

- **Write_lock(x)**
  - B : if lock(x) = ―unlocked‖ then begin
    - lock(x) ―write_locked‖ else if
    - lock(x) = ―write_locked‖
    - wait (until lock(x) = ―unlocked‖) else begin
    - lock(x)=―read_locked‖ then
      - wait (until lock(x) = ―unlocked‖)

    end;
- **Unlock(x)**
- If lock(x) = ―write_locked‖ then Begin
- Lock(x) ―unlocked‖
- Else if
- lock(x) = ―read_locked‖ then Begin
- No_of_read(x) no_of_read(x) - 1
- If (no_of_read(x) = 0) then Begin
- Lock(x) ―unlocked‖ End

Conn…

Thank You…….