TRANSACTION & CONCURRENCY CONTROL

-Anil Shrestha (070/BCT/01)
-Bijay Sahani (070/BCT/05)
-Bimal Shrestha (070/BCT/10)
-Deshbhakta Khanal (070/BCT/13)

17 March 2017

What is a Transaction?

- Logical unit of work in database
  - An entire program
  - A portion of program
  - A single command
- Must be either entirely completed or aborted
- No intermediate steps are acceptable
- Successful transactions change database from one consistent state to another

ACID Properties Of Transaction

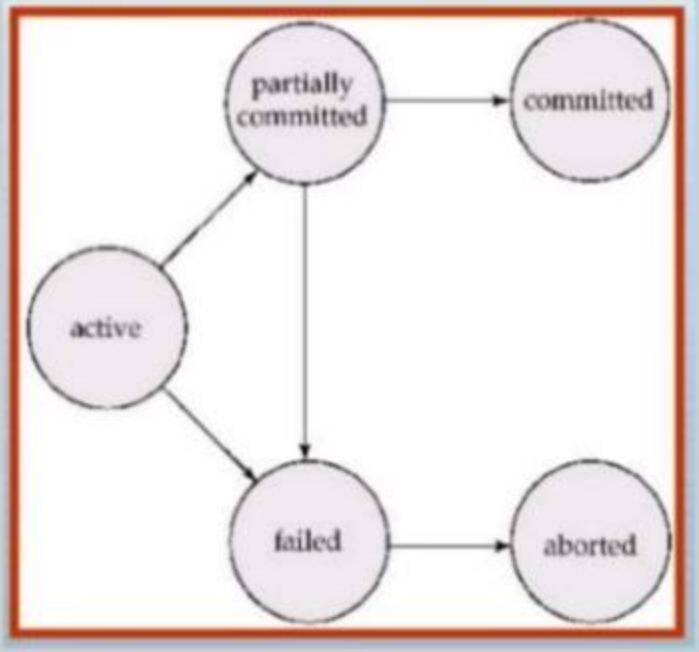
- Atomic – All or nothing
  - All parts of transaction must be completed and committed or must be aborted and rolled back
- Consistent
  - Conducts transactions in serial order
  - Important in multi-user and distributed databases

ACID Properties Of Transaction

- Isolation
  - There may be execution of multiple transaction and each transaction must be unware of each other
- Durability
  - Changes made to database persists even if there are system failures

States Of Transaction

- Active
- Partially Committed
- Committed
- Failed
- Aborted



Nested Transaction

- A transaction that is created inside another transaction
- While the nested (child) transaction is active, the parent transaction may not perform any operations other than to commit or abort, or to create more child transactions
- When a parent aborts, all of its sub-transactions are aborted
- When a sub-transaction aborts, parent can decide whether to abort or not

Lock

- A lock guarantees exclusive use of a data item to a current transaction
- To access data item (lock acquire)
- After completion of transaction(release lock)
- All data item must be accessed in mutual exclusive manner

Types of Lock

- Shared lock
- Exclusive lock

Shared Lock

- When we want read the data item value at that time shared lock is used
- Lock-s

Exclusive Lock

- It is used for both read and write
- Lock-x

Compatibility Between Lock Modes

	shared	exclusive
shared	true	false
exclusive	false	false

Clip slide

Example of lock

T1	T2
Lock-x(B)	
R(B)	
B-50	
W(B)	
Unlock(B)	
	lock-s(B)
	R(B)
	unlock(B)

Optimistic Concurrency Control

- Optimistic concurrency control does not involve locking rows when reading
- Optimistic concurrency control (OCC) helps increase database performance
- Fewer resources are used to hold the locks during the update process
- Records are locked for a shorter length of time

Method For Concurrent Control

- Deadlock:
  - A state in which each member of a group of transactions is waiting for some other member to release a lock.
- Prevention:

  Lock all the objects used by a transaction when it starts → not a good way.

Method For Concurrent Control

- Drawbacks of locking:

  Lock maintenance represents an overhead that is not present in systems that do not support concurrent access to shared data.

- Deadlock:

  Deadlock prevention reduces concurrency

Method For Concurrent Control

- Timestamp ordering:
  - Each transaction is assigned a unique timestamp values when it starts
  - Timestamp defines its position in the time sequence of transaction

Method For Concurrent Control

- Timestamp ordering write rule:

if ($T_c \geq$ maximum read timestamp on D &&
$T_c >$ write timestamp on committed version of D)
perform write operation on tentative version of D with write timestamp T_c
else */*write is too late*/*
abort transaction T_c

Method For Concurrent Control

- Timestamp ordering read rule:

If (T_c > write timestamp on committed version of D)

{ let $D_{selected}$ be the version of D with the maximum write timestamp ≤ Tc

if ($D_{selected}$ is committed)

perform read operation on the version $D_{selected}$

else

wait until the transaction that made version $D_{selected}$ commits or aborts then reapply the read rule

}

Else

abort transaction T_c

Method For Concurrent Control

- Multi-version timestamp ordering:
  - A list of old committed versions as well as tentative versions is kept for each object.
  - Read operations that arrive too late need not be rejected.