CSE2004 – Database Management Systems

Text Books :

1.R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7 th Edition, 2015

2.Raghu Ramakrishnan, Database Management Systems,Mcgraw-Hill,4th edition,2015

Reference Book

3.A. Silberschatz, H. F. Korth & S. Sudershan, Database System Concepts, McGraw Hill, 6th Edition 2010

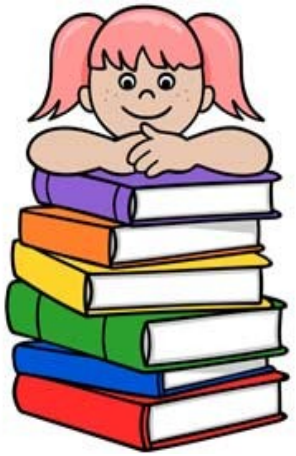
4.Thomas Connolly, Carolyn Begg," Database Systems : A Practical Approach to Design, Implementation and Management",6th Edition,2012

5.Shashank Tiwari ,"Professional NoSql",Wiley ,2011

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Unit – 5 : Concurrency Control and Recovery Techniques

- ✓ **Introduction to Concurrency Control**
  - ➢ **Two-Phase Locking**
  - ➢ **Control based on time-stamp**
- ✓ **Introduction to Recovery Concepts**
  - ➢ **Recovery based on deferred update**
  - ➢ **Recovery based on immediate update**
  - ➢ **Shadow paging**



Introduction to Concurrency Control

- In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions.
- We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions.
- Concurrency control protocols can be broadly divided into two categories –
    - Lock based protocols
    - Time stamp based protocols

Introduction to Concurrency Control

1 Purpose of Concurrency Control

- To enforce Isolation (through mutual exclusion) among conflicting transactions.
- To preserve database consistency through consistency preserving execution of transactions.
- To resolve read-write and write-write conflicts.

Example:

In concurrent execution environment if T_1 conflicts with T_2 over a data item A, then the existing concurrency control decides if T_1 or T_2 should get the A and if the other transaction is rolled-back or waits.

Introduction to Concurrency Control

Concurrency Control Protocols : It is a set of rules, that guarantee the serializability, to ensure the isolation property.

Two-phase locking protocols : technique of locking data items to prevent multiple transactions from accessing the items concurrently. (*used in Commercial DBMS, but high Overhead*)

Timestamps : It is an Identifier, generated by system, and its value are generated in the same order as the transaction start times to ensure the serializability and assumes Multi-version of data item exists.

Optimistic Protocols : based on concept of validation or certification of a transaction after it executes its operations along with multi-version;

Snapshot isolation(Optimistic Protocols): are used in a number of commercial DBMSs and in certain cases are considered to have lower overhead than locking-based protocols.

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Lock based protocols :

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds

- Binary Locks(Mutual Exclusion) –
  - A lock on a data item can be in two states; it is either locked or unlocked.
- Shared/exclusive –
  - If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state.
  - Read locks are shared because no data value is being changed.

Introduction to Concurrency Control

There are four types of lock protocols available −

- Simplistic Lock Protocol :
  - obtain a lock on every object before a 'write' operation is performed.
  - unlock the data item after completing the ‘write’ operation.
- Pre-claiming Lock Protocol :
  - Used evaluate their operations and create a list of data items on which they need locks.
  - Before initiating an execution, the transaction requests the system for all the locks it needs.
  - If all the locks are granted, the transaction executes and releases all the locks when all its operations are over.
  - If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.

Introduction to Concurrency Control

There are four types of lock protocols available −

- Two-Phase Locking 2PL :
  - It divides the execution phase of a transaction into three parts.
  - In the first part, when the transaction starts executing, it seeks permission for the locks it requires.
  - The second part is where the transaction acquires all the locks.
  - In third phase, the transaction cannot demand any new locks; it only releases the acquired locks.

In 2PL locking has two phases,

Growing : where all the locks are being acquired by the transaction; and
Shrinking : where the locks held by the transaction are being released.

To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Introduction to Concurrency Control

There are four types of lock protocols available −

- Strict Two-Phase Locking :
  - The first phase of Strict-2PL is same as 2PL.
  - After acquiring all the locks in the first phase, the transaction continues to execute normally.
  - But in contrast to 2PL, Strict-2PL does not release a lock after using it.
  - Strict-2PL holds all the locks until the commit point and releases all the locks at a time.

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Two-Phase Locking Techniques

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.

Two Phase Locking Techniques(Components)

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Lock Manager: Managing locks on data items.

Lock table: Lock manager uses it to store the *TransactionID*, *Data item, lock mode* and pointer to the next data item locked.

One simple way to implement a lock table is through linked list.

Transaction ID	Data item id	lock mode	Ptr to next data item
T1	X1	Read	Next

Database requires that all transactions should be well-formed.

A transaction is well-formed if:

- It must lock the data item before it reads or writes to it.
- It must not lock an already locked data items and

it must not try to unlock a free data item.

Two Phase Locking Techniques(Binary Lock)

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

lock_item(X):

B: if LOCK(X) = 0
(*item is unlocked*)
then LOCK(X) ← 1 (*lock the item*)
else
begin
wait (until LOCK(X) = 0 and the lock manager wakes up the transaction);
go to B
End;

unlock_item(X):

LOCK(X) ← 0; (* unlock the item *)
if any transactions are waiting then wake up one of the waiting transactions;

Two Phase Locking Techniques
(Shared/Exclusive Lock)

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Algorithm for read_lock(X):

```
B:  if LOCK(X) = "unlocked"
            then begin      LOCK(X) ← "read-locked";
                            no_of_reads(X) ← 1
                 end
            else if LOCK(X) = "read-locked"
                 then no_of_reads(X) ← no_of_reads(X) + 1
                 else  begin
                          wait (until LOCK(X) = "unlocked"  and
                           the lock manager  wakes up the transaction);
                       go to B
                  end
    end
```

Two Phase Locking Techniques
(Shared/Exclusive Lock)

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Algorithm for write_lock(X):

```
B:  if LOCK(X) = "unlocked"
        then LOCK(X) ← "write-locked"
        else
            begin
                wait (until LOCK(X) = "unlocked"
                and the lock manager wakes up the transaction);
            go to B
            end
        end
```

Two Phase Locking Techniques
(Shared/Exclusive Lock)

Algorithm for unlock(X):

```
if LOCK(X) = "write-locked"
then
        begin
                LOCK(X) ← "unlocked";
                wakeup one of the waiting transactions, if any
        end
    else if LOCK(X) = "read-locked"
            then begin
                no_of_reads(X) ← no_of_reads(X) –1;
                if no_of_reads(X) = 0
                then begin LOCK(X) = "unlocked";
                        wakeup one of the waiting transactions, if any
                     end
            end
    end;
```

Two Phase Locking Techniques
(Upgrade/Downgrade Lock)

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Lock conversion(Upgrade/Downgrade)

Lock upgrade: *Existing read lock to write lock*

if T_i has a read-lock(X) and T_j has no read-lock(X) (i # j) then

convert read-lock (X) to write-lock (X)

else

force T_i to wait until T_j unlocks X

Lock downgrade: *Existing write lock to read lock*

Ti has a write-lock (X) (*no transaction can have any lock on X*)

convert write-lock (X) to read-lock (X)

Two Phase Locking Techniques

Guaranteeing Serializability

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

T_1	T_2
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
unlock(Y);	unlock(X);
write_lock(X);	write_lock(Y);
read_item(X);	read_item(Y);
X := X + Y;	Y := X + Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

Initial values: X=20, Y=30

Result of serial schedules :

- $T_1 \leftarrow T_2$: X=50, Y=80
- $T_2 \leftarrow T_1$: X=70, Y=50

No 2PL is applied because

- write_lock(X) operation follows the unlock (Y) operation in T_1 , and
- write_lock(Y) operation follows the unlock (X) operation in T_2

Two Phase Locking Techniques

Guaranteeing Serializability

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

T_1	T_2
read_lock(Y);	
read_item(Y);	
unlock(Y);	
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	Y := X + Y;
	write_item(Y);
	unlock(Y);
write_lock(X);	
read_item(X);	
X := X + Y;	
write_item(X);	
unlock(X);	

Initial values: X=20, Y=30

Result of Nonserial schedules :

- X=50, Y=80

Two Phase Locking Techniques

Guaranteeing Serializability

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

T_1	T_2
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
write_lock(X);	write_lock(Y);
unlock(Y)	unlock(X)
read_item(X);	read_item(Y);
X := X + Y;	Y := X + Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y)

Initial values: X=20, Y=30

Result of Nonserial schedules :

- X=?, Y=?

2PL is applied

Dealing with Deadlock and Starvation

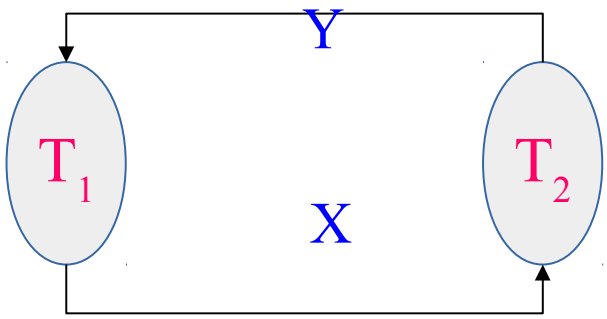
Deadlock : occurs when each transaction T in a set of two or more transactions is waiting for some item that is locked by some other transaction T′ in the set.

Hence, each transaction in the set is in a waiting queue, waiting for one of the other transactions in the set to release the lock on an item, But because the other transaction is also waiting.

For Eg :

T_1	T_2
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Wait-Graph



Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Timeouts : (low overhead and simplicity)

- If a transaction waits for a longer period than a system-defined timeout period, the system assumes that the transaction may be deadlocked and aborts it, *regardless of whether a deadlock actually exists*.

Starvation :

- When a transaction cannot proceed for an indefinite period of time, while other transactions in the system continue normally.
- This may occur if the waiting scheme for locked items is unfair in that it gives priority to some transactions over others.
- One solution for starvation is to have a fair waiting scheme, such as using a FCFS queue; transactions are enabled to lock an item in the order in which they originally requested the lock.
- Another scheme allows some transactions to have priority over others but increases the priority of a transaction the longer it waits, until it eventually gets the highest priority and proceeds.

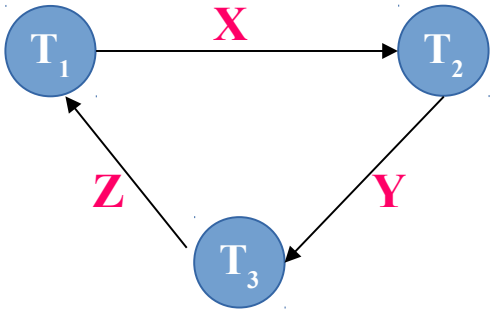
Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Deadlock Handling in Centralized Systems : There are 3 classical approaches

- Deadlock prevention.
- Deadlock avoidance.
- Deadlock detection and removal.

All three approaches, incorporated in both a centralized/distributed databases.



Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Deadlock Prevention :

- Does not allow any transaction to acquire locks that will lead to deadlocks.
- i.e., when more than one transactions request for locking the same data item, only one of them is granted the lock.

One of the popular deadlock prevention methods is pre-acquisition of all the locks.

- In this method, a transaction acquires all the locks before starting to execute and retains the locks for the entire duration of transaction.
- If another transaction needs any of the already acquired locks, it has to wait until all the locks it needs are available.
- Using this approach, the system is prevented from being deadlocked since none of the waiting transactions are holding any lock

Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Deadlock Avoidance :

- It handles deadlocks before they occur.
- It analyzes the transactions and the locks to determine whether or not waiting leads to a deadlock.

The method can be briefly stated as follows.

- Transactions start executing and request data items that they need to lock.
- The lock manager checks whether the lock is available. If it is available, the lock manager allocates the data item and the transaction acquires the lock.
- However, if the item is locked by some other transaction in incompatible mode, the lock manager runs an algorithm to test whether keeping the transaction in waiting state will cause a deadlock or not.
- Accordingly, the algorithm decides whether the transaction can wait or one of the transactions should be aborted.

Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Deadlock Avoidance :

- There are two algorithms for this purpose, namely wait-die and wound-wait.

Let us assume that there are two transactions, T_1 and T_2, where T_1 tries to lock a data item which is already locked by T_2.

The algorithms are as follows −

- Wait-Die − If T_1 is older than T2, T_1 is allowed to wait. Otherwise, if T_1 is younger than T_2, T1 is aborted and later restarted.

- Wound-Wait − If T_1 is older than T_2, T_2 is aborted and later restarted. Otherwise, if T_1 is younger than T_2, T_1 is allowed to wait.

Dealing with Deadlock and Starvation

Not to have Deadlock in a System

Deadlock Detection and Removal

- It runs a deadlock detection algorithm periodically and removes deadlock in case there is one.
- It does not check for deadlock when a transaction places a request for a lock.
- When a transaction requests a lock, the lock manager checks whether it is available. If it is available, the transaction is allowed to lock the data item; otherwise the transaction is allowed to wait.
- Since there are no precautions while granting lock requests, some of the transactions may be deadlocked.
- To detect deadlocks, the lock manager periodically checks if the wait-forgraph has cycles. If the system is deadlocked, the lock manager chooses a victim transaction from each cycle.
- The victim is aborted and rolled back; and then restarted later.

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Deadlock Detection and Removal

Some of the methods used for victim selection are −

- Choose the youngest transaction.
- Choose the transaction with fewest data items.
- Choose the transaction that has performed least number of updates.
- Choose the transaction having least restart overhead.
- Choose the transaction which is common to two or more cycles.

Case Study : Deadlock Handling in Distributed Systems

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Timestamp Ordering Protocol :

- Ensures serializability among transactions in their conflicting read and write operations.
- Conflicting pair of tasks should be executed according to the timestamp values of the transactions.
    - The timestamp of transaction T_i is denoted as $TS(T_i)$.
    - Read time-stamp of data-item X is denoted by R-timestamp(X).
    - Write time-stamp of data-item X is denoted by W-timestamp(X).

Time-Stamp based Concurrency Control

- The use of locks, with the 2PL protocol, guarantees serializability of schedules. (i.e., based on the order in which executing transactions, lock the items they acquire).
- If a transaction needs an item that is already locked, it may be forced to wait until the item is released.
- Some transactions may be aborted and restarted because of the deadlock problem.
- A different approach that guarantees serializability involves using transaction time-stamps to order transaction execution for an equivalent serial schedule.
- Time-stamp : A unique identifier, created by DBMS to identify a transaction, whose values are assigned in the order in which the transactions are submitted to the system(denoted as TS(T)).

Time-Stamp based Concurrency Control

- Timestamp ordering (TO) : Schedule in which the transactions participate is then serializable, and the only equivalent serial schedule permitted has the transactions in order of their timestamp values.
- Notice how this differs from 2PL, where a schedule is serializable by being equivalent to some serial schedule allowed by the locking protocols
- In timestamp ordering, however, the schedule is equivalent to the particular serial order corresponding to the order of the transaction timestamps.

Time-Stamp based Concurrency Control

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

- The algorithm must ensure that, for each item accessed by conflicting operations in the schedule, the order in which the item is accessed does not violate the timestamp order.
- The algorithm associates with each database item X two timestamp (TS) values:

1. read_TS(X) : The read timestamp of item X is the largest timestamp among all the timestamps of transactions that have successfully read item X. i.e., *read_TS (X) = TS (T)*, where T is the youngest transaction that has read X successfully.
2. write_TS(X) : The write timestamp of item X is the largest of all the timestamps of transactions that have successfully written item X. i.e.,*write_TS (X) = TS (T),* where T is the youngest transaction that has written X successfully.

Time-Stamp based Concurrency Control

Basic TimeStamp Ordering

1. Transaction T issues a write_item(X) operation:
   - If read_TS(X) > TS(T) or if write_TS(X) > TS(T), then an younger transaction has already read the data item so abort and roll-back T and reject the operation.
   - If the condition in part (a) does not exist, then execute write_item(X) of T and set write_TS(X) to TS(T).
2. Transaction T issues a read_item(X) operation:
   - If write_TS(X) > TS(T), then an younger transaction has already written to the data item so abort and roll-back T and reject the operation.
   - If write_TS(X) ≤ TS(T), then execute read_item(X) of T and set read_TS(X) to the larger of TS(T) and the current read_TS(X).

Time-Stamp based Concurrency Control

Strict Timestamp Ordering

1. Transaction T issues a write_item(X) operation:
   - If TS(T) > read_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).

2. Transaction T issues a read_item(X) operation:
   - If TS(T) > write_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).

Time-Stamp based Concurrency Control

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Thomas's Write Rule

- If read_TS(X) > TS(T) then abort and roll-back T and reject the operation.
- If write_TS(X) > TS(T), then just ignore the write operation and continue execution. This is because the most recent writes counts in case of two consecutive writes.
- If the conditions given in 1 and 2 above do not occur, then execute write_item(X) of T and set write_TS(X) to TS(T).