CS 344 Lecture Notes

Date : 2 November 2011

Topic : Transaction Concurrency Control Schemes

Prepared By:

1. Ankit Jaiswal 09010108
2. Sunil Ratnu 09010155

There are four important **properties of transactions** that a DBMS must ensure to maintain data in the face of concurrent access and system failures:

1. Atomicity 2.Consistency 3.Isolation 4.Durability

Ques) Is it always the case that having 2 Parallel Transactions are better than having only 1 Transaction and having 3 Transaction instead of 2 is better and so on?
Ans) No, it is not always the case. Increasing the number of Transactions increases the Resource Utilization upto a certain limit only.

Having many transactions with only many Read Values is not a Problem but problem arises when they involve write operations as well. So, how DBMS decides how many transactions to run in Parallel? One very easy way is to put just a number as a limit on the number of parallel transactions, but still, that would not solve the problem of simultaneous reads and writes. DBMS follows certain methodology to resolve the problem of Concurrency Control which are discussed below:

Lock Based Protocol

Suppose there are 2 transactions T1 and T2 such that:
T1 reads value A and then write back A & T2 first reads A, then reads B and finally write back B.

T1	**T2**
R_A	R_A
W_A	R_B
	W_B

To maintain concurrency as well as to avoid inconsistency a table can be maintained for all the transactions :

Value	A	
Type of Access	W	
Transaction	T1	

Any transaction requests the DBMS before accessing any resource. If any other transaction is using the same resource then the new transaction may not proceed further. For the above case, when a new transaction T2 comes, then DBMS will check the above table, if the same resource is already in process then T2 won't proceed and will wait.

A particular lock is acquired by a transaction while it executes and the lock is released when the transaction completes its work. A new transaction involving the same locked resource can execute simultaneously if both the transactions involve read operations only.

Lock / New Transaction→	**R**	**W**
R	Allowed	Not Allowed
W	Not Allowed	Not Allowed

In the above case deadlocks can occur. For example, consider the two transactions :
T1 [W_A and then W_B] and T2 [W_B and then W_A]

Order of Execution: 1. T1 : W_A 2. T2 : W_B 3. T1 : W_B 4. T2 : W_A

Value	A	B	
Type of Access	W	W	
Transaction	T1	T2	

After executing first and second step i.e. 1. T1 : W_A , 2. T2 : W_B , the write locks for A and B are acquired which get released when T1 & T2 both ends but it never happens because for T1 to finish it has to write A and then write B and to write B it requires the write lock (already acquired by T2) which never get released as T2 has to yet write A (already acquired by T1) to end its execution.
Lazy Solution for this : Don't check for deadlocks.

2-Phase Locking Protocol

It involves two phases:
1. Growing Phase –Locks are acquired and no locks are released.
2. Shrinking Phase – Locks are released and no locks are acquired.

Simple 2-phase Locking :
All locks are acquired in the Growing Phase and released in the Shrinking Phase. No lock is released until all the locks are acquired.

Strict 2-phase Locking :
All locks are acquired in Growing phase and only Read locks can be released in the Shrinking Phase. All Write locks are released at the completion of the Transaction.

Rigorous 2-phase Locking :
All locks are acquired in the Growing Phase and all locks are released only at the completion of the transaction.

Simple 2-phase locking has the **better Resource utilization** and **lesser probability of Deadlocks** as compared to the other 2-phase locking protocols.
Deadlocks can't be removed completely. Only the Probability can be minimized.

Cascading Roll Back :
Suppose transaction T1 releases some locks in the Shrinking Phase and those locks get acquired by T2 and T3. If T1 fails before its completion, then the values used by T2 and T3 also becomes invalid and they also have to stop the execution and undo all the changes. This is known as **Cascading Roll Back.**
No Cascading Roll Back occurs in Rigorous 2-phase locking Protocol.

Different Levels of Locking:

1. Cell based Locking
2. Row based Locking
3. Column based Locking
4. Table based Locking

In case of table based locking, there is worst resource utilization with fewer number of Locks and low probability of Deadlocks.

Other Schemes for Concurrency Controls :

One way can be to create copies of the resources and make them available to all the transactions. Hence, no transaction waits and no deadlocks occur. But in this case many Write values will be there for a single resource value. This can be sorted out by following the below strategies :

1. One who commits first wins
2. One who requests first wins.