Distributed Database System (CS-600)

Mr. Nauman Iqbal email id: nauman@biit.edu.pk Whatsapp# 0321-5821151
Mr. Ahsan email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656

(Week 11) Lecture 21-22 (Chapter 20 of Book)

Assignemnt#9 (given at End)

1. **Submission Date: Sunday 10-May-2020 before 11:59PM.**
2. **You must prepare handwritten Assignment and send it to respective course teacher (after scanning it) for assessment by email only.**

Assignment submission through Email:
Email Subject must be in Correct format otherwise assignment will not be checked.
Email Subject Format: 4digitReg-DDS-section-ASG8

Example: 1234-DDS-CS6A-ASG3 Correct
2017-ARID-1234-DDS-CS6A-ASG3 Incorrect

Objectives: Learning objectives of this lecture are

- **Drawback of Two-Phase Locking**
- **Deadlock**
- **Deadlock prevention Techniques**
  - **Time-Out**
  - **Wait-Die**
  - **Wound-Wait**
  - **Conservative Two-Phase Locking**

Overview:

In this lecture we will study drawback of two-phase locking, what are deadlocks, what are techniques of deadlock prevention, how to detect a deadlock in a schedule and recovery from deadlock detection.

Recap:

Previously, we have studied that transaction must be executed at the same time so that no transaction must wait to get executed. But the problem arises that transaction executed concurrently may cause concurrency problems which leads to the inconsistent database. Afterwards, we studied locking mechanism. Locking is used to avoid concurrent access on same data-item. That is multiple transaction will be executed concurrently but cannot access same data-item at the same time. But problem occurred in locking that arithmetic operator whose precedence is different causes inconsistency in database. Then we moved toward two-phase locking which removes the problem occurred in locking. But two-phase locking also has a problem that results in ultimate WAIT situation for multiple transactions which in result seize the system at a certain point.

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

Problems in Two-Phase Locking:

Two-Phase locking says "no lock operation must appear after first unlock operation in a transaction". It removes consistency problems but There is a problem that is caused by Two-phase locking. Let us examine it by example.

Ta	Tb
Begin-transaction	
Lock(balx)	
Read(balx)	**Begin-Transaction**
balx=balx+100	Lock(baly)
Write(balx)	Read(baly)
Lock(baly)	baly=baly+50
WAIT	Write(baly)
WAIT	Lock(balx)
WAIT	WAIT
WAIT	WAIT

- As we know, Lock operation is request for data-item from DBMS. Before achieving Lock, a transaction cannot issue another operation and has to wait for lock to be released by other transaction.
- So both above transactions have come to state where both are waiting and cannot process further.
- Above state of both transactions will result into unlimited wait.

Deadlock:

An impasse that may result when two (or more) transactions are each waiting for locks to be released that are held by the other.

A system is said to be in Deadlock when

- Transaction (A) is holding a lock on data-item(x), and waiting for lock to be allocated on data-item(y).
- Transaction (B) is holding a lock on data-item(y), and waiting for lock to be allocated on data-item(x).
- In this case each transaction is holding a data-item and each is waiting for one-another for a lock to be release.

Deadlock seizes a system and stop it from further execution. Which causes unlimited wait which is never wanted in any system. A resource must not be occupied for a very large amount of time by any transaction so that other transaction may access the resource and system must not stop working.

Distributed Database System (CS-600)

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

Example 1:

Saad and Saman are siblings. They have a couple of toys to play with.
A writing board and a board marker. The problem is each of one has one item.

- Saad has got board marker and Saman has writing board.
- Now both are them are asking each other to give them their toy but both are refusing.
- So they have to WAIT for each other unless one of them back-off.
- Now both are in situation of "Deadlock".

Example 2:

Now Consider the given schedule.

Ta	Tb
Begin-transaction	
Lock(balx)	
Read(balx)	**Begin-Transaction**
balx=balx+100	Lock(baly)
Write(balx)	Read(baly)
Lock(baly)	baly=baly+50
WAIT	Write(baly)
WAIT	Lock(balx)
WAIT	WAIT
WAIT	WAIT

- Two transaction Ta and Tb are executing in Non-Serial mode of execution.
- Ta is holding (balx) and waiting for (baly).
- Tb is holding (baly) and waiting for (balx).
- Both transactions are holding a data-item which is required by each other. This is called a deadlock.
- System cannot move further unless WAIT is over. But this WAIT is unlimited.

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

Solution to Deadlock:

There is only one solution to deadlock after it has occurred in system and that is "Kill any One". Which means one transaction has to back-off to end deadlock.

Deadlock Prevention Algorithms:

However, there are certain algorithms which prevents/ends deadlock. Some of them are given below.

1- **Time-Out**
2- **Wait-Die**
3- **Wound-Wait**
4- **Conservative 2PL**

1- Time-Out Algorithm:

According to time-out algorithm, whenever a transaction requests for a lock on data-item, and lock is not allocated to transaction. In this case, a timer will be associated with the transaction. For a specific period of time, if lock is not allocated to a transaction then transaction will be considered in deadlock (maybe there is no deadlock) and it will be killed(rollback) by system.

NOTE: Max waiting time depends upon system and can be adjusted.

Example1:

Now let us use the above schedule as an example and see how time-out works. Here max waiting time is set to **3** time operations. After 3 time operations of WAIT, transaction will be killed by system. And all of its locks will be released.

Time	Ta	Waiting Timer Ta	Tb	Waiting Timer Ta
t0	**Begin-transaction**			
t1	Lock(balx)			
t2	Read(balx)		**Begin-Transaction**	
t3	balx=balx+100		Lock(baly)	
t4	Write(balx)		Read(baly)	
t5	Lock(baly)		baly=baly+50	
t6	WAIT	wait-1	Write(baly)	
t7	WAIT	wait-2	Lock(balx)	
t8	WAIT	wait-3	WAIT	wait-1
t9	**Rollback**	Rollback Ta	Read(balx)	

Here we can see,

- Ta's first WAIT operation is at time t6, and waiting timer starts at t6 for transaction Ta.
- Tb's first WAIT operation is at time t8, and waiting timer starts at t8 fro transaction t8.
- At time t8, max wait time of transaction Ta is completed. Max wait time is set to 3 operations.
- As max wait time is completed system killed transaction Ta at time t9.
- As transaction Ta is killed(rollback) all of its locks are unlocked. And assigned to Tb which has requested for balx.

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

Conclusion:

Deadlock in above schedule is prevented using timestamping technique. Ta will be Rollback at time t9.

Example2:

Given is another example of how time-out works.

Time	Ta	Waiting Timer Ta	Tb	Waiting Timer Ta
t0	**Begin-transaction**			
t1	Lock(balx)			
t2	Read(balx)		**Begin-Transaction**	
t3	balx=balx+100		Lock(baly)	
t4	Write(balx)		Read(baly)	
t5	Lock(baly)		baly=baly+50	
t6	WAIT	wait-1	Write(baly)	
t7	WAIT	wait-2	Lock(z)	
t8	WAIT	wait-3	Read(z)	
t9	**Rollback**	Rollback Ta	z=z+50	
			Write(z)	
			Commit	

We can see there is no deadlock in above schedule. Because Tb is not waiting for any other transaction. All lock required by Tb are allocated. But on the other hand, Ta required baly which is occupied by Tb. As max waiting time period of Ta has expired, **system has rollback the transaction Ta regardless of Deadlock. This is drawback of time-out algorithm.**

2- Wait-Die Algorithm

It states that **"Older transaction will wait for the lock on data-item, Younger will be killed on requesting lock on data-item held by an older transaction".**

- ✓ If older transaction will ask for a lock which is held by a younger transaction, older will wait until lock is released by the transaction or transaction is completed.
- ✓ If Younger will request a lock on data-item which is held by an older transaction, then younger transaction will be kill(rollback) by system.

- Older Transction
  Which appears first in the system.
- Younger Transaction
  Which appears after another transaction.

Example of Younger/Older Transaction:

Time	Ta	Tb	Tc	Td
t0		Begin-Transaction		Begin-Transaction
t1	Begin-Transaction			
t2			Begin-Transaction	

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

- Priority will be given from left to right
- Given is the numbering of transaction oldest to youngest.

1- Tb (Oldest)
2- Td
3- Ta
4- Tc (Youngest)

Example1:

Time	Ta	Tb	Action
t0	**Begin-transaction**		
t1	Lock(balx)		
t2	Read(balx)	**Begin-Transaction**	
t3	balx=balx+100	Lock(baly)	
t4	Write(balx)	Read(baly)	
t5	Lock(baly)	baly=baly+50	
t6	WAIT	Write(baly)	
t7	WAIT	Lock(balx)	**Tb will be Rollback**
t8	Read(baly)		
t9	baly=baly+100		
t10	Commit		

- Ta is Older transaction and Tb is Younger.
- Ta requires baly at time t5, but baly is assigned by younger Tb so Ta has to wait.
- Tb requires balx at time t7, but balx is assigned to an Older Transaction Ta so now Tb will be killed(rollback) by system.
- As Tb is killed, all of its lock are unlocked so baly will be assigned to Ta preventing deadlock.

Example 2:

Time	Ta	Tb	Action
t0	**Begin-transaction**		
t1	Lock(balx)	**Begin-Transaction**	
t2	Read(balx)	Lock(baly)	
t3	balx=balx+100	Read(baly)	
t4	Write(balx)	baly=baly+50	
t5	Lock(balz)	Write(baly)	
t6	balz=balz*2	Lock(balx)	Tb will be Rollback
t7	Write(balz)		
t8	Commit		

In above example we can see that there is no deadlock as Ta is not waiting for any data-item. Besides that, according to Wait-Die algorithm, whenever younger transaction will ask for a lock from an older transaction, younger has to die. That is why in above transaction, Tb is killed by system. **This is drawback of Wait-Die algorithm.**

3- Wound-Wait Algorithm

It states that **"Younger transaction will have to wait if a lock on data-item is held by an older transaction but Older transaction will kill younger transaction if it requires a lock on data-item that is held by a younger transaction".**

- ✓ Younger will wait if lock is held by older.
- ✓ Older will kill younger if lock is held by younger.

Example 1:

Time	Ta	Tb	Action
t0	**Begin-transaction**		
t1	Lock(balx)	**Begin-Transaction**	
t2	Read(balx)	Lock(balz)	
t3	balx=balx+100	Read(balz)	
t4	Write(balx)	balz=balz / 5	
t5	.	Lock(balx)	
t6	.	WAIT	
t7	Lock(balz)	WAIT	**Tb will be Rollback**
t8	Read(balz)		
t9	baly=balz*2		
t10	Write(baly)		
t11	Commit		

- Tb is younger transaction, when it required balx which is held by and Older transaction, it has to wait.
- Ta is Older transaction, when it required balz which is held by a younger transaction Tb, younger is killed at time t7.

Example 2:

Time	Ta	Tb	Action
t0	**Begin-transaction**		
t1	Lock(balx)		
t2	Read(balx)	**Begin-Transaction**	
t3	balx=balx+100	Lock(baly)	
t4	Write(balx)	Read(baly)	
t5	Lock(baly)		**Tb will be Rollback**
t6	Read(baly)		
t7	baly=baly+100		
t8	Commit		

- Here we can see there is no deadlock because Tb does not need any data-item which is held by an older transaction.
- But when Ta which is older transaction required baly which is held by a younger transaction Tb, according to Wound-Wait algorithm, Tb has to be killed.

4- Conservative 2-Phase Locking

- A transaction must be allocated all locks before it starts.
- In this way transaction will not have to wait due to which no deadlock will occur.

Drawback:

Mr. Nauman Iqbal email id: nauman@biit.edu.pk Whatsapp# 0321-5821151
Mr. Ahsan email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656

- It is hard to gain all lock at the same time specially when more locks are required by a transaction.
- In this case, transaction may start after too much delay or even never delay because of non-availability of all locks at the same time.

Example 1:

Let us say that there are we are going on a trip towards northern areas. But before starting our journey, we will make a list of important things to take with us. Let us say we will take following important things without which we may get in big trouble

1- 50,000 Rs Cash
2- Vehicle Puncture Kit
3- Enough fuel

Now if we apply Conservative two-phase locking, we will first gather all above resources and then we will start our journey.
Advantage of this algorithm is that we have got all the important things we needed once we have started our journey, now during our journey we don't have to wait for any of the resource to be allocated.
On the other hand, if we do not take all resources and forget cash amount to take with us. It may cause us real trouble and we have to wait for our friends to bring us some cash so that we can move forward.

Example 2:

Given are three transactions that appear in the system. Each transaction will first ask DBMS to allocate all of the resources(data-items) it requires. If all resources are allocated to a transaction than it will begin other-wise it will have to wait for all the resources to be allocated.

Transaction	Resources Required
T1	bal-x, bal-y, bal-z
T2	bal-t, bal-x, balq
T3	bal-s, bal-r

As Transaction T1 appears first so it will ask DBMS to allocated bal-x, bal-y and bal-z to it. As there is no other transaction appeared so it will be allocated all of its required resources. Now major point is that transaction T1 will not wait for any of other transaction because it has been allocated all of its resources.

Afterwards when T2 will appear in the system, it will request DBMS to allocate bal-t, bal-x and bal-q to it. But as bal-x is already been allocated to transaction T1 so transaction T2 will wait for transaction T1 to be completed to get access towards bal-x. so T2 will not even start.
Afterwards when T3 will appear in the system, it will request DBMS to allocate bal-s and bal-r. As these data-items are not allocated to any other transaction so these will be assigned to transaction T3. And T3 will not wait for any other transaction.

Conclusion: As there is not wait in the execution of transaction so there are no chances of deadlock to appear.

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

Assignment # 9

Question No 1: Consider the given schedule and tell what **transaction** will be killed **at first** if following algorithms are used to prevent deadlock. (also write operation and time at which transaction will be killed, for time-Out algorithm; after two operations transaction will be killed). Also list transaction names in descending order with respect to their age.

i- wound-wait **ii- wait-die** **iii- timeOut**

Time	T1	T2	T3	T4	T5	T6
t_0			**begin tran**	**begin tran**	**begin tran**	
t_{01}	**begin tran**	**begin tran**	lock(y)		lock(z)	
t_{02}	lock(x)	lock(t)	y=y*2		z=z+50	
t_{03}	x=x+10	t=t-20	y=y+100	lock(s)	write(z)	**begin tran**
t_{04}	write(x)	lock(b)	lock(c)	read(s)		lock(e)
t_{05}	lock(a)	read(b)	read(c)	s=s+150	lock(q)	read(e)
t_{06}	read(a)	lock(e)	lock(z)	lock(t)	read(q)	e=e+44
t_{07}	a=a+50	read(e)	read(z)	read(t)	q=q*5	lock(a)
t_{08}	commit	e=e+80	z++	t=t+s	lock(t)	read(a)
t_{09}		commit	write(z)	commit	read(t)	a=e+50
t_{10}			commit		commit	Commit

Explanation:
Above is given a schedule which have six transactions. Here Wait is not mentioned because this is the list of operations exists in each transaction. You have to examine above schedule carefully and have to see which transaction will be allocated lock on data-item and which will be denied. Lock operation appearing at first will be given data-item and later will have to wait.

In Question number 1, you have to do following tasks.

1- List all the names of transaction in descending order with respect to their age. For example, at number 1, the name of oldest transaction must appear, and at number 6, the name of youngest transaction must appear.
2- Three deadlock prevention algorithms are mentioned, you have to go through the schedule carefully and just write name of first transaction that will be killed by the system according to
   a. **Wait-Die Algorithm**
   b. **Wound-Wait Algorithm**
   c. **Time-Out Algorithm**

Mr. Nauman Iqbal **email id: nauman@biit.edu.pk Whatsapp# 0321-5821151**
Mr. Ahsan **email id: ahsan@biit.edu.pk, Whatsapp# 0333-5189656**

You must mention the time stamp (t0 to t10) and the operation due to which transaction is killed.

Also mention name of transaction which cause the death of the transaction. You must write a brief reason.

Question No 2: Solve given lab manuals number 9 given with lecture notes.