<u>Distributed Database System (CS-600)</u>

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

(Week 10) Lecture 10-11 (Chapter 20 of Book)

<u>Assignemnt#8</u> (given at End)

1. **Submission Date: Sunday 3-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**

- **Locking Protocol**
- **Types of Locks**
- **How to Apply Locking Protocol**
- **Two Phase Locking**

Overview:

In this lecture, we will learn about locking protocol. Why it is applied? What are Locks? Types of Locks. What can be achieved by applying locking. Drawbacks of Locking protocol. Two phase Locking.

Recap:

In last lecture, we have learnt problems in concurrent transactions. And we have agreed that a transaction must execute in a concurrent way to avoid waiting and transaction must be accurate so that database must be consistent.

Now we are going to study a protocol that will make sure consistency in concurrent transactions.

Locking:

A procedure used to control concurrent access to data. When one transaction is accessing the database, a lock may deny access to other transactions to prevent incorrect results.

Types of Locks:

1- **Shared lock:**
If a transaction has a shared lock on a data item, it can read the item but not update it.

2- **Exclusive lock:**
If a transaction has an exclusive lock on a data item, it can both read and update the item.

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

Locking Procedure:

- Any transaction that needs to access a data item must first lock the item, requesting a shared lock for read only access or an exclusive lock for both read and write access.
- If the item is not already locked by another transaction, the lock will be granted.
- If the item is currently locked, the DBMS determines whether the request is compatible with the existing lock. If a shared lock is requested on an item that already has a shared lock on it, the request will be granted; otherwise, the transaction must **wait** until the existing lock is released.

Summary:

Lock Held (Ta)	Lock Requested (Tb)	Action
None	Shared	Lock Granted
None	Exclusive	Lock Granted
Shared	Shared	Lock Granted
Shared	Exclusive	Lock Denied – Wait
Exclusive	Shared	Lock Denied – Wait
Exclusive	Exclusive	Lock Denied – Wait

A transaction continues to hold a lock until it explicitly releases it either during execution or when it terminates (aborts or commits). It is only when the exclusive lock has been released that the effects of the write operation will be made visible to other transactions.

Note: (We will consider all lock as Exclusive locks in our transactions)

Example:

Ta	Tb
Begin-Transaction	
Lock(x)	**Begin-Transaction**
Read(x)	Lock(y)
x = x + 10	y = y + 50
Lock(y)	y = y * 2
WAIT	Write(y)
WAIT	Commit
Read(y)	
y = y + 30	
Write(y)	
Commit	

- Lock operation is request to DBMS to grant a transaction lock over any data-item.
- If lock is not assigned to any other transaction then it will be granted, otherwise transaction has to wait until lock is released from the transaction or transaction is completed.

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

- We can See, Transaction Ta requests for (x) to be granted, as (x) is not allocated to any of the transaction in the schedule so it is granted.
- Read(x) means lock has been granted to the transaction. Any transaction will only read data when lock is granted.
- Transaction Tb has requested for Lock on data-item (y), as it is not allocated to any of transaction so it is allocated to Tb.
- Transaction Ta has requested for Lock on data-item (y), but as data-item (y) is already been assigned to Transaction Tb, so lock is not allocated to Transaction Ta. So Transaction Ta has to wait until Transaction Tb releases lock or completed.
- As Transaction Tb Issues Commit statement, Lock for data-item (y) is allocated to Transaction Ta and it reads data-item (y).

Applying Locking on Problems Occurred in Concurrency Control:

1- Lost Update Problem

Time	**T1**	**T2**	**balx**
t1		begin-transaction	**100**
t2	begin-transaction	read(balx)	**100**
t3	read(balx)	balx = balx + 100	**100**
t4	balx = balx -10	write(balx)	**200**
t5	write(balx)	commit	**90**
t6	commit		**90**

Before applying locking, value of balx is 90 which is incorrect. 190 is correct value.

- Now Applying Locking protocol on above problem.

Time	**T1**	**T2**	**Balx**
t1		begin-transaction	100
t2	begin-transaction	Lock(x)	100
t3	Lock(x)	read(balx)	100
t4	WAIT	balx = balx + 100	100
t5	WAIT	write(balx)	200
t6	WAIT	commit	200
t7	read(balx)		200
t8	balx = balx -10		200
t9	write(balx)		190
t10	commit		190

Here We Can See, after applying LOCKING protocol, value in balx is accurate which is 190.

2- Uncommitted dependency (or dirty read) problem:

Time	**T1**	**T2**	**balx**
t1		begin-transaction	**100**
t2		read(balx)	**100**

---

t3		balx = balx + 100	**100**
t4	begin-transaction	write(balx)	**200**
t5	read(balx)	⋮	**200**
t6	balx = balx -10	rollback	**100**
t7	write(balx)		**190**
t8	commit		**190**

Before applying LOCKING protocol, value of balx is “190” which is incorrect. Accurate value of balx is “90” in this case.

- Now applying LOCKING protocol.

Time	**T1**	**T2**	**balx**
t1		begin-transaction	**100**
t2		Lock(balx)	**100**
t3		read(balx)	**100**
t4	begin-transaction	balx = balx + 100	**200**
t5	Lock(balx)	write(balx)	**200**
t6	WAIT	⋮	**100**
t7	WAIT	rollback	**100**
t8	read(balx)		**100**
t9	balx = balx -10		**100**
t10	write(balx)		**90**
t11	commit		**90**

- Here We can see, after applying LOCKING protocol, we have accurate value in balx which is “90”.

3- Inconsistent analysis problem:

Time	**T5**	**T6**	**balx**	**baly**	**balz**	**sum**
t1		begin-transaction	100	50	25	
t2	begin-transaction	sum=0	100	50	25	0
t3	read(balx)	read(balx)	100	50	25	0
t4	balx = balx – 10	sum = sum + balx	100	50	25	100
t5	write(balx)	read(baly)	90	50	25	100
t6	read(balz)	sum = sum + baly	90	50	25	150
t7	balz = balz + 10	.	90	50	25	150
t8	write(balz)	.	90	50	35	150
t9	commit	read(balz)	90	50	35	150
t10		sum = sum + balx	90	50	35	185
t11		commit	90	50	35	185

Value of sum is 185 here which is incorrect, it must be 175 as sum of balx(90), baly(50) and balz(35) is 175.

- Now applying LOCKING protocol on above schedule.

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

Time	**T5**	**T6**	**balx**	**baly**	**balz**	**sum**
t1		begin-transaction	100	50	25	
t2	begin-transaction	sum=0	100	50	25	0
t3	Lock(balx)	Lock(balx)	100	50	25	0
t4	read(balx)	WAIT	100	50	25	0
t5	balx = balx – 10	WAIT	100	50	25	0
t6	write(balx)	WAIT	90	50	25	0
t7	UnLock(balx)	read(balx)	90	50	25	0
t8	Lock(Balz)	sum = sum + balx	90	50	25	90
t9	read(balz)	Lock(baly)	90	50	25	90
t10	balz = balz + 10	read(baly)	90	50	25	90
t11	write(balz)	sum = sum + baly	90	50	35	140
t12	commit	.	90	50	35	140
t13		.	90	50	35	140
t14		Lock(balz)	90	50	35	140
t15		read(balz)	90	50	35	140
t16		sum = sum + balz	90	50	35	175
t17		commit	90	50	35	175

- Here We can see, after applying LOCKING protocol, we have accurate value in sum variable which is “175”.

Problem in Locking Protocol:

As we have seen above, all three problems occurred due to concurrent transaction have been resolved using Locking protocol. But there is still a problem exists in Locking. Let us see it using an example.

Time	**Ta**	**Tb**	**x**	**y**
t01	Begin-transaction		100	50
t02	Lock(x)	Begin-transaction	100	50
t03	Read(x)	Lock(y)	100	50
t04	x=x+50	Read(y)	100	50
t05	Write(x)	y=y*2	150	50
t06	Unlock(x)	Write(y)	150	100
t07	Lock(y)	Lock(x)	150	100
t08	WAIT	Read(x)	150	100
t09	WAIT	Unlock(y)	150	100
t10	Read(y)	x=x+100	250	130
t11	y=y+30	Write(x)	250	130
t12	Write(y)	Commit	250	130
t13	Commit		250	130

Now as we know, Serial schedule are always correct. Now let us execute it as a Serial schedule and then as a Non Serial schedule and see the difference.

Schedule	x	y
Serial	250	160

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

Non-Serial	250	130

- As value of 'y' is not same in serial and non-serial schedule which means above schedule is not serializable.
- Above problem has occurred due to precedence of an operator.
- In serial schedule, 30 is added in y first and then it is multiplied by 2.
- In non-serial schedule, y is multiplied by 2 first and then 30 is added.

Two-phase locking (2PL):

A transaction follows the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction.

Once transaction has issued an 'Unlock' statement, then it will not issue 'Lock' statement for any of the data-item. Which means all lock operations will appear before first 'Unlock' operation is issued. Or it can also be said that 'after first Unlock operations issued, transaction cannot Lock any data-item'.

According to the rules of this protocol, every transaction can be divided into two phases:

- **Growing phase**: in which it acquires all the locks needed but cannot release any locks,
- **Shrinking phase:** in which it releases its locks but cannot acquire any new locks.

Now applying Two-phase locking protocol on above problem which occurred due to simple locking protocol.

Time	Ta	Tb	x	y
t01	Begin-transaction		100	50
t02	Lock(x)		100	50
t03	Read(x)		100	50
t04	x=x+50		100	50
t05	Write(x)		150	50
t06	Lock(y)		150	50
t07	Read(y)		150	50
t08	Unlock(x)	Begin-transaction	150	50
t09	y=y+30	Lock(y)	150	50
t10	Write(y)	WAIT	150	80
t11	Commit	Read(y)	150	80
t12		y=y*2	150	80
t13		Write(y)	150	160
t14		Lock(x)	150	160
t15		Read(x)	150	160
t16		x=x+100	150	160
t17		Write(x)	250	160
t18		Commit	250	160

- Here we have seen, value of x=250 and y=160 which is correct.

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

Questions that may appear in one's mind.

1- Locking solves the consistency but it causes wait. Then why don't we just use Serial-Schedule?

Ans: In serial-schedule, transaction has to wait for previous transaction to complete regardless of conflict statements or data-item. Where as in Locking, it only forces other transaction to wait where there is a conflict between each other. Otherwise transaction will not have to wait.

2- Two-phase locking is just like serial schedule.

Ans: Our first priority is to make database consistence, so to achieve this target we have to sacrifice a little bit.

Problems in Two-Phase Locking:

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.

Assignment # 8

Question#1: Why locking is mandatory for ensuring consistency? [explain briefly in your own words with example]

Question#2: Why is Two-Phase locking preferred over simple Locking? Explain with example of your own. [explain briefly in your own words with example]