JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-11, November 2018 E-ISSN: 2347-2693

A study of Preventing Concurrency's Problems using 2-Phase Locking Protocols (2-PL)

Anil Kumar Singh

Dept. of Information Technology, Jagran Institute of Management, Kanpur -208 014, India

[*]*Corresponding Author: anil.sysadmin@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 18/Nov/2018, Published: 30/Nov/2018

Abstract- Now days every organization has importance of time in its day to day working. To save time, the task should be executed in distributed manner and also to adopt the procedure to perform, parallel or concurrent execution method. Concurrency means, more than one transactions are performing at the same time then they are interleaving to each other. When the transactions are inter-leaving for short span they cause the different types of problems like lost update, dirty read etc. To control these types of problems, there are several methods like Locking Methods, Time-stamp Methods and Optimistic Methods. In this paper we will study the 2-Phase Locking Protocol which comes under locking method. With the help of 2-PL, we shall reveal how to prevent the problem arise due to concurrency with the help of suitable examples. It will help the students and research scholars to understand that how to prevent the concurrency problems with the help of 2-Phase Locking Protocol (2-PL) method.

Keywords- 2-PL, Growing, Shrinking, Locks, Concurrency, Dirty Read, Lost Update, inconsistent analysis

I. INTRODUCTION

When a data base is retrieved and updated simultaneously by a number of running transactions it could lead to inconsistencies in the stored or retrieved data if such operations are not regulated. [1] Ensuring the reliability of the stored/retrieved data is an important issue in database management when the database is accessed and updated parallel by a number of transactions.

A common tactic to this problem is to define a transaction as a unit that preserves consistency, and require that the outcome of processing a set of transactions at the same time as be the same as one produced by running these transactions serially in some order [2].

It is the process of managing simultaneous execution of transactions in a shared database, to ensure the serializability. The main objectiveof concurrency control**s:**

a. To enforce isolation: Means one transaction is not interleaving with another transaction.
b. To preserve database consistency
c. To solve read-write and write-write conflicts

Though 2-phase locking is sufficient to preserve the serializability of transactions, it is not sufficient to guarantee isolation, because a transaction can release its exclusive locks at any time during the shrinking phase and a different transaction can therefore observe its result before its commitment.[3]

The paper is organized as follows:
Section I contains the Introduction of Concurrency and objective of concurrency control in Distributed Database System, Section II Contains lock and their types and compatibility between Locks, Section III contains introduction of 2-Phase Locking Protocols and 2PL graph. In this section problems are explained and their justification is given with the help of suitable example, Section IV contains problems with concurrency, Section V contains method of preventing concurrency problems. Section VI concludes the research work.

II. LOCKS

A lock guarantees exclusive use of a data item to a concurrent transaction.To access a data item locks are require. After completion of transaction the lock should be released. All data items must be accessed in a mutually exclusive manner. It means if one transaction is accessing the data items no other transaction is allowed to update that data item simultaneously.
Types of lock: Basically locks are two types i.e. shared lock and exclusive lock.
Shared lock: Lock – S used for Read
Exclusive lock: Lock – X used both read and write

Compatibility between lock models

Lock	**Shared – S**	**Exclusive -X**
Shared – S	Yes	No
Exclusive -X	No	No

Any number of transactions can hold shared lock but exclusive lock can be hold only by one transaction at a time.

Example of lock protocols

Transaction-1	Transaction-2
Lock –S (y)	Shared Lock
Read (y)	
Unlock (y)	
Exclusive Lock	Lock-X (y)
	Read (y)
	y:=y-20
	Write (y)
	Unlock (y)

III. TWO PHASE LOCKING PROTOCOL (2-PL)

2-Phase Locking Protocol is one of the concurrency control technique. It is one in which there are 2 phases that a transaction goes through. The first is the **Growing Phase**in which it is only acquiring locks not releasing the locks, the second is the **Shrinking Phase**in which it is releasing locks. Once you have released a lock, you cannot acquire any newer locks. The two phase locking rule basically states that no transaction should request for a lock after it release one of its locks. On the other hand a transaction should not release lock until it is sure that it will not request for another new lock.[4] Because it has two phases i.e. Growing Phase (Acquiring the Locks), Shrinking Phase (Releasing Locks). Hence it is called 2-Phase Locking Protocol. This protocolensures a serializable schedule.In between Growing Phase and Shrinking Phase there is a common point known as lock point.

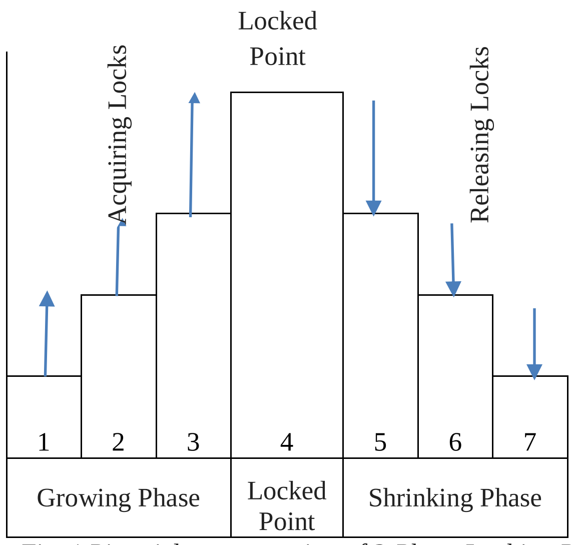

Fig. 1 Pictorial representation of 2-Phase Locking Protocol

Example of 2-Phase Locking protocols Schedule -S

T1	T2
Lock –X (A)	
Read (A)	
A=A+50	
Write (A)	
Lock-X (B)	
Unlock (A)	
	Lock-X (A)
	Read (A)
	A:=A*10
	Write (A)
	Unlock (A)
Read (B)	
B=B+20	
Write(B)	
Unlock(B)	
	Lock-X (B)
	Read (B)
	B=B*10
	Write (B)
	Unlock (B)

Serializable Schedule-S

IV. PROBLEMS WITH CONCURRENCY TRANSACTIONS

The problems are given below:

A. Lost update problem
B. Uncommitted dependency or dirty read problem
C. Inconsistent analysis problem [5].

V. METHOD OF PREVENTING CONCURRENCY PROBLEMS

A. Lost update problem – one older transaction is doing some update on a certain variable but the younger transaction update will cancel out, nullified the update make by the older transaction.

Preventing lost update problem using 2 Phase Locking Protocol (2PL)

Tran.-1	**Time**	**Tran.-2**	**Sum**
	t_1	Begin_tran	1000
Begin_tran	t_2	Lock-X (Sum)	1000
Lock-X (Sum)	t_3	Read (Sum)	1000
Wait	t_4	Sum:=Sum+100	1000
Wait	t_5	Write (Sum)	1100
Wait	t_6	Commit	1100
Wait	t_7	Unlock(Sum)	1100
Read(Sum)	t_8		1100

Sum=Sum - 50	t_9		1100
Write(Sum)	t_{10}		1050
Commit	t_{11}		1050
Unlock (Sum)	t_{12}		1050

In the above example Tran-2 is originated at t_1 time stamp and Tran.-1 is originated at t_2 time stamp. Hence Tran-2 will be treated as older transaction and Tran-1 will be treated as younger transaction.
Tran. -2 which is the older transaction locks the data item "Sum" by the executable lock, the younger transaction Tran. -1 also applies to lock the "Sum" data variable on t_3 time stamp. But the permission will not be granted because "Sum" is already locked by Transaction Tran.-2.
Now younger transactionTran. -1 will wait for lock until the older transaction release the lock. The older transaction reads the value of data item sum =1000 and it updates the value by increasing to value by 100 on t_4 time stamp.
Finally, older transaction Tran.-2 is updates the value of sum=1100 and release the lock. Now request of transaction Tran.-1 will be granted for locking the data item "Sum". The younger transaction acquires the lock and it reads the value of sum =1100. Tran.-1 updates the value of sum by decreasing to valve by 50 on t_9 time stamp and the resultant value of sum=1050. In this way we can prevent the lost update problem

B. Uncommitted dependency or dirty read problem- when a transaction is uses the result of partially completed transaction.

Preventing uncommitted dependency or dirty read problem using 2 Phase Locking (2PL)

Tran.-1	Time	Tran.-2	Sum
	t_1	Begin_tran	1000
Begin_tran	t_2	Lock-X (Sum)	1000
Lock-X (Sum)	t_3	Read (Sum)	1000
Wait	t_4	Sum:=Sum+100	1000
Wait	t_5	Write (Sum)	1100
Wait	t_6	rollback	1000
Wait	t_7	Unlock(Sum)	1000
Read(Sum)	t_8		1000
Sum=Sum - 50	t_9		1000
Write(Sum)	t_{10}		950
Commit	t_{11}		950
Unlock (Sum)	t_{12}		950

In the above example Tran-2 is originated at t_1 time stamp and Tran.-1 is originated at t_2 time stamp. Hence Tran-2 will be treated as older transaction and Tran-1 will be treated as younger transaction.
Tran. -2 which is the older transaction locks the data item "Sum" by the executable lock, the younger transaction Tran. -1 also applies to lock the "Sum" data variable on t_3 time stamp. But the permission will not be granted because "Sum" is already locked by Transaction Tran.-2.
Now younger transaction Tran.-1 will wait for lock until the older transaction release the lock. The older transaction reads the value of data item sum =1000 after that Tran-2 rollbacks and unlock the sum variable.
Now request of transaction Tran.-1 will be granted for locking the data item "Sum". The younger transaction acquires the lock and it reads the old value of sum i.e. 1000 instead of 1100 because older transaction Tran-2 rolled back before the commit operation. Tran.-1 updates the value of sum by decreasing to valve by 50 on t_9 time stamp and the resultant value of sum=950. In this way we can prevent the uncommitted dependency or dirty read problem.

C. Inconsistent analysis problem- When a transaction reads several values from the database but a second transaction update some of them during the execution of the first. One transaction is doing some processing with the values of some variables fixed from the database and concurrently another transaction is doing some updates on those values. Hence it is known as inconsistent analysis problem.

Preventing inconsistent analysis problem using 2Phase Locking (2PL)

Tran-1	**T_s**	**Tran-2**	**A**	**B**	**C**	**Sum**
	t_1	Begin_tran	200	100	50	
Begin_tran	t_2	Sum=0	200	100	50	0
Lock-X (A)	t_3		200	100	50	0
Lock-X (C)	t_4	Lock-S (A)	200	100	50	0
Read(A)	t_5	Wait	200	100	50	0
A=A-10	t_6	Wait	200	100	50	0
Write(A)	t_7	Wait	190	100	50	0
Read (C)	t_8	Wait	190	100	50	0
C=C+10	t_9	Wait	190	100	50	0
Write (C)	t_{10}	Wait	190	100	60	0
Commit	t_{11}	Wait	190	100	60	0
Unlock (A,C)	t_{12}	Wait	190	100	60	0
	t_{13}	Lock-S (B)	190	100	60	0
	t_{14}	Lock-S (C)	190	100	60	0
	t_{15}	Read (A)	190	100	60	0
	t_{16}	Sum=Sum+ A	190	100	60	0
	t_{17}	Write(Sum)	190	100	60	190
	t_{18}	Read(B)	190	100	60	190
	t_{19}	Sum=Sum+ B	190	100	60	190
	t_{20}	Write(Sum)	190	100	60	290
	t_{21}	Read(C)	190	100	60	290
	t_{22}	Sum=Sum+ C	190	100	60	290
	t_{23}	Write(Sum)	190	100	60	350

	t_{24}	Commit	190	100	60	350
	t_{25}	Unlock(A,B, C)	190	100	60	350

In the above example Tran-2 is originated at t_1 time stamp and Tran.-1 is originated at t_2 time stamp. Hence Tran-2 will be treated as older transaction and Tran-1 will be treated as younger transaction.

Tran-1 which is the younger transaction locks the data items A, C by exclusive locks at t_3, t_4 time stamp respectively. Tran-2 request for lock the data item "A" at time stamp t_4 but the permission will not be granted because data item "A" is already locked by Tran-1.

Hence the Tran-2 will wait for lock until the Tran.-1 release the lock. Tran.-1 updates the value of A by decreasing to value by 10 on t_6 time stamp and the resultant value of A=190. It also updates the value of C by increasing to value by 10 on t_9 time stamp and the resultant value of C=60. Finally it releases the lock of A, C at time stamp t_{12}. Tran-2 locks the data items B, C at time stamp t_{13}, t_{14} respectively.

The value of "Sum" updates by Tran-1 by adding the value of "A". Sum=Sum+A, Sum=0+190, Sum=190 at time stamp t_{16}. The value of "Sum" further updates by Tran-2 by adding the value of B. Sum= Sum+B, Sum=190+100, Sum=290 at time stamp t_{19}. Again the value of "Sum" updates by transaction Tran-2 by adding the value of date item C. Sum=Sum+C, Sum=290+60, Sum=350 at time stamp t_{22}.

Finally, we got the values of "Sum" as 350. In this way we can prevent the inconsistence analysis problem by using 2PL.

VI. CONCLUSION

Concurrency means, more than one transaction are performing at the same time. When transactions are performing simultaneously they causes the problems like lost update, dirty read and inconsistence analysis etc. Ensuring the reliability of the stored/retrieved data is an important issue in database management when the database is accessed and updated parallel by a number of transactions. It is found that with the help of 2-Phase Locking Protocols we may prevent the concurrency and the problems arise due to the concurrency i.e. lost update, dirty read and inconsistence analysis etc.

The limitation of this paper is that I have studied only 2PL to prevent concurrency and their problems in distributed database systems. In future we may work on another methods of concurrency control such as Time-stamp Methods, Optimistic Methods. We may also do comparatively study among them.

REFERENCES

[1]. Kedeml C.et. al., "An Efficient Deadlock Removal Scheme for Non-Two-Phase LockingProtocols", Proceedings of the Eighth International Conference on Very Large Data Bases, Mexico City, September, 1982.

[2]. C. MOHAN et. al., "Lock Conversion in Non-Two-Phase Locking Protocols", IEEE Transactions on Software Engineering, vol. se-11, No. 1, p.p. 15-22, January 1985

[3]. Stefano Ceri, Giuseppe Pelagatti, " Distributed Database Principles and Systems", Tata McGraw-Hill Edition, ISBN 0-07-026511-9, p.p. 195

[4]. M. Tamer Ozsu, Patrick Valduriez, "Principles of Distributed Database Systems",Pearson Education, 2nd Editions, ISBN 81-7758-177-5, p.p. 262 , 2008

[5]. Singh Anil Kumar, "A study of Concurrent transaction execution and their problems in Distributed Database System", International Journal ofComputer Sciences and Engineering, Volume-6 , Issue-10, Page no. 810-813, ISSN 2347-2693(E), Oct-2018

AUTHOR'S PROFILE

Dr. Anil Kumar Singh is Postgraduate in Computer Science, M.Sc., MCA, PGDCA (1st Position in CSJM University, Kanpur) and Doctorate in Information Technology and having a large 18 years' experience in Academic and Research. Dr. Singh has presented and published more than 20 papers in various National and International Journals and Conferences. His area of expertise is in Computer Network, Database Management, RDBMS, Wi-Fi Technology, Cyber Security, Client/Server Computing, Linux, CISCO and Ethical Hacking. He is a professional life member of Indian Science Congress Association etc. Dr. Singh has a vast experience in academic field and served as Head Computer Center in Dr. GHS-IMR, Kanpur for more than 4 years and presently working as Associate Professor and Academic Head in JIM, Kanpur since year 2005. Dr. Singh has participated various workshops and Short Term Courses organised by the prestigious institutes like I.I.T., Kanpur, I.I.T., Delhi and various technical universities.
Moreover, he has organised various workshops related with Computer Networking, Security and Ethical hacking. Dr. Singh Chaired in the Technical Sessions of International Conferences like IEEE, ICSPICC2016, organized by SSBT College of Engineering, Jalgaon, Maharashtra, IEEE, 2nd ICCIT, organized by Siddhant College of Engineering, Pune, Maharashtra.