Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.1, May 2016 – www.caeaccess.org

Survey on Concurrency Control Techniques

Marwa Mohamed
Computer Science & Eng. Dept.,
Faculty of Electronic Eng.,
Menoufia University, Menouf
32952, Egypt

Mohammed Badawy
Computer Science & Eng. Dept.,
Faculty of Electronic Eng.,
Menoufia University, Menouf
32952, Egypt

Ayman El-Sayed
Computer Science & Eng. Dept.,
Faculty of Electronic Eng.,
Menoufia University, Menouf
32952, Egypt

ABSTRACT
The coordination of the simultaneous execution of transactions in a multiuser database system may violates with consistency, performance and correctness of whole database. Data must maintain in consistency with correctness concurrency control techniques try to make balance between these characteristics and take in account time and performance of transactions. In this paper we had discussed various concurrency techniques, their advantages and disadvantages and make comparative study between them

General Terms
Concurrency, Algorithms

Keywords
Concurrency, locking, optimistic, multi version, performance

1. INTRODUCTION
When user begins its transaction for specific request the database state is changed. In any individual transaction, which is running in isolation, is assumed to be correct [11]. or when there are several transactions are executing on different data items at the same time also it is assumed to be correct while in shared database several transactions are executes concurrently in the database specified in same data item and at the same time the isolation property may no longer be preserved [11]. If transactions are executed serially, i.e., sequentially with no overlap in time, no transaction concurrency exists [3]. However, if concurrent transactions with interleaving operations are allowed in an uncontrolled manner, some unexpected, undesirable result may occur, such as The lost update problem, The dirty read problem and The incorrect summary problem [3], The lost update problem occurs when two concurrent transactions are updating the same data element and one of the updates is lost (overwritten by the other transaction) [21], The dirty read problem Transactions read a value written by a transaction that has been later aborted. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read"). The reading transactions end with incorrect results [3], and the incorrect summary problem [3]. While one transaction takes a summary over the values of all the instances of a repeated data-item, a second transaction updates some instances of that data-item. The resulting summary does not reflect a correct result for any (usually needed for correctness) precedence order between the two transactions (if one is executed before the other), but rather some random result, depending on the timing of the updates, and whether certain update results have been included in the summary or not. Process of managing simultaneous execution of transactions in a shared database, to ensure the Serializability of transactions, is known as concurrency control [17]. To ensure that the system must control the interaction among the concurrent transactions; this control is achieved through one of a variety of mechanisms called concurrency control techniques such locking based methods, timestamp based method, multi version based methods and optimistic method. The serializable transactions are executed one at a time, or serially, rather than concurrently [20].

In order to evaluate the performance of concurrency control techniques, there are some evaluation metrics such as:

1. Accuracy
Defines the percent at which correctness of data can be achieved, after commitment of transaction updates that may violates correctness of data.

2. Serializability
[21] Ensures that the schedule for the concurrent execution of the transactions yields consistent results and transaction can be executed in the same order of sending their request.

3. Number of Committed, Wait and Rollback Transactions
Defines number of committed transactions (transactions that executed successfully and commit their updates), number of wait transactions (transactions that still wait their order to commit their updates) and number of rollback transactions (transactions that aborted from system due to occurrence of conflict with committed one) good technique should provide high number of committed transactions than wait one and lowest number of rollback transactions.

4. Deadlock
[21] Occurs when two transactions wait indefinitely for each other to unlock data [11].

5. Storage
Memory and RAM requirements for database storage and versions created from it.

This paper is organized as follow: we provide taxonomy of concurrency control techniques, and the advantages and disadvantages for each class of solution for concurrency problem. In section III, we discuss the open points. Finally, we conclude this survey in section IV.

2. TAXONOMY OF CONCURRENCY CONTROL TECHNIQUES
The serializable transactions are executed one at a time, or serially, rather than concurrently [4]. In this paper we intent to compare the following concurrency control techniques: (1) Lock-Based Protocols, (2) Two-Phase Locking Protocol, (3) Timestamp-Based Protocols, (4) Multi version Schemes, (5) Optimistic Protocols.

2.1 Lock-Based Protocols

Principles- A lock guarantees exclusive use of a data item to a current transaction. Lock-Based Protocols maintain lock of data item to only one transaction while others prevented from access to locked item until current transaction release its lock from data item then it becomes free for others. Data items can be locked in two modes; either exclusive (X) mode or shared mode (S). [1] For transactions that can both read and write from the data item X, exclusive-mode lock is given. For transactions that can read, but cannot write on item S, shared-mode lock is given to data item. Transaction can proceed only after request is granted. [11] N number of transactions can hold shared locks (S) on an item. But if any transaction holds an exclusive lock (X) on the item, no other transaction may hold any lock on that item. In this condition, a lock cannot be granted and the requesting transaction has to wait until all incompatible locks held by other transactions are released. The lock is then granted. [1], [8], [14], [19].

Discussion- there are two problems of lock based protocols are the resulting transaction schedule might not be serializable and the schedule might create deadlock.

2.2 Two-Phase Locking Protocol

Principles- This Protocol used to ensure seriabilzability by forcing some restriction on transaction as Any transaction is not allowed to obtain new locks till it had released a lock this restriction called two phase locking(2pl). This protocol called 2pl because it has two principal phases as in figure (1). The first phase is known as the growing phase; in which a transaction acquires all the locks it needs. The second phase is known as the shrinking phase, where the process releases the locks. [1] If a process fails to acquire all the locks during the first phase, then it is obligated to release all of them, wait, and then start over. [12] This protocol ensures conflict–serializable schedules. Such described in [1], [2], [3], [14-15].

Discussion- This protocol may be good in case of absence of any information about the transactions or the database. There are two types of two phase locking protocol as in figure (1): strict two-phase locking and rigorous two-phase locking.

2.2.1 Strict two-phase Locking

In this protocol any transaction does not release any of its exclusive write lock until after it commits or aborts. Other transactions cannot access to locked item until current transaction has committed. Transaction must hold all its exclusive locks till it commits or aborts and no cascading rollback takes place. Read lock of transaction can be released when transaction terminates (commits its results) but write must be maintained until after commitment or abortion of transaction.

2.2.2 Rigorous two-phase Locking

It is more restrictive variation of strict 2PL. all locks (read and write) are held until after transaction commits or aborts. Drawbacks of these protocols are starvation which occurs when a transaction cannot proceed for an indefinite period of time while other transactions in the system continue normally.

2.3 Timestamp-Based Protocols

Principles- This protocol is used to keep information about the precise of the order of arrival of execution. Locking algorithms are ignored for this protocol instead this algorithm is implemented using timestamps. Timestamp is a unique value that is assigned to transaction when it begins. Also each data item has write timestamp (WTS) and read timestamp (RTS). WTS is the largest time stamp of transaction that execute write operation successfully. RTS is the largest time stamp of transaction that execute read operation successfully. The protocol manages concurrent execution using time-stamps to determine the Serializability order. Try to ensure Serializability by taking priority to transaction with lower timestamp (older transaction) to access data before other transactions with higher timestamp. when process wants to access data, timestamp protocol checks transaction's timestamp and read and write timestamp for data item, if RTS and WTS of data older than it for transaction, transaction read or write process complete successfully, else transaction has to abort. Such discussed in [4], [5], [14-16].

Discussion- From advantage of This protocol is that it solves problem of appearance of deadlock .as in this protocol each data item in database has two values for timestamp, one for the last time the field was read and one for the last update, this increases memory needs and the database's processing overhead.

2.4 Multi version concurrency control Schemes

Principles- This protocol keeps the old values of a data item when the item is updated, there are number of versions of data item assigned for transaction for write operation and right version is maintained for read operation. when transaction issue write operation, it writes a new version and old version is retained. In this paper we will discuss two multi version protocols (Multi version Two-Phase Locking, Multi version Timestamp Ordering) as in figure 2 such discussed in [7], [14], and [18-19].

Discussion- An obvious drawback of multi version concurrency techniques is that more storage is needed for multiple versions of data.

2.4.1 Multi version Timestamp Ordering

In this protocol the timestamps are used to label the version. When a read operation is issued, an appropriate version of data based on the timestamp of the transaction is selected, and the value of the selected version is returned. Reads never have to wait as an appropriate version is returned immediately [1].

When a transaction issues a write step on some entity Y, we might choose not to overwrite the old value of Y by the new one, but to keep both versions. If subsequently another transaction reads Y, we have the option of supplying to it either version, whichever serves Serializability best, as that is the final accepted action. In this scheme, each data item Y has a sequence of versions <Y1, Y2,...., Yn>.

Each version of data contains three data fields, one for data value, one for write timestamp that equal to timestamp of latest transaction that created wrote version of data successfully, and one for read timestamp that equal to s largest timestamp of a transaction that successfully read version. Conflict will occur if transaction wants to write the same version that is currently read by another transaction, write operation cannot succeed. Such described in [10], [14], [18-19].

2.4.2 Multi Version to Phase using Certify Lock

In this multiple-mode locking scheme, there are three locking modes for an item: read, write, and certify, instead of just the two modes (read, write) discussed previously. Hence, the state

of LOCK(X) for an item X can be one of read-locked, write-locked, certify-locked, or unlocked [14].

In this multi-version 2PL scheme, reads can proceed concurrently with a single write operation—an arrangement not permitted under the standard 2PL schemes. The cost is that a transaction may have to delay its commit until it obtains exclusive certify locks on all the item sit has updated [14].

This scheme can be powerful as it avoids cascading aborts, since transactions are only allowed to read the version X that was written by a committed transaction. but deadlocks may occur if upgrading of a read lock to a write lock is allowed.

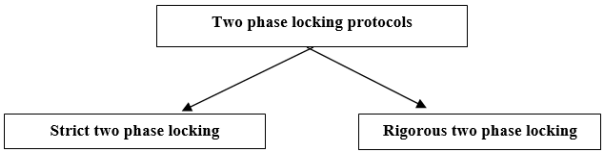

Fig 1: Types of Two phase locking protocols

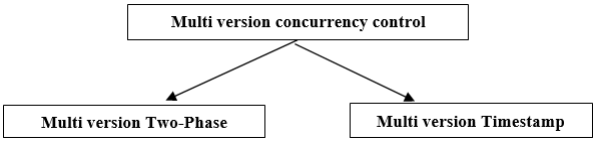

Fig 2: Classification of multi version concurrency control

2.5 Optimistic Protocols

Principles- In all previous concurrency control techniques, checking is done before operation execution, as in locking protocol check to determine if data item is locked also in timestamp check transaction timestamp against read and write timestamp of data item. In optimistic concurrency control, that is also called validation based technique, check is done after read phase and before write phase to minimize overhead during transaction execution, with the effect of slowing down the transactions. Here transaction passes with three phases, read phase in which transaction can read committed data from database, validation phase that check whether update done by transaction will not affect database consistency, write phase if validation success transaction updates are applied to database else updates will be discarded and transaction is restarted, such as [2], [3], [5], [6], [13], [14], and [19].

Discussion- This protocol may be good algorithm in the situations that conflict will be rare to occur. Optimistic scheme, we do not lock the records and therefore no deadlocks occur.

3. OPEN POINTS AND DISCUSSION

We have so far described and compared a large variety of proposals. Finally, this paper shows some points that help researchers in this subject to improve and provide new and good technique for solving concurrency problem.

This paper now discusses some key points that can be modified according to the exact application requirements.

- ***TRANSACTION RATE***: transaction can be defined as a sequence of information exchange and related work that is treated as a unit for the purposes of satisfying a request and for ensuring database integrity. In concurrency control subject. transaction rate can be defined as number of transactions that sending request for data at the same time.
- ***USER_COMMITS_SEC***: as user can execute several operations and produce several change for data stored in database, these change may take some time for commitment, performance of system may depend on this time as if time user take to commit its update is too small this will improve from system performance so metric called user_commits_sec means number of commitment of updates applied by user per second. ***USER_ROLLBACKS_TXN***: as probability of conflict in concurrency system may be high so there are number of transactions should be aborted or rolled back to solve conflict between two or more transactions this can be measured using metric USER_ROLLBACKS_TXN that represent number of times system aborts the same transaction.
- ***Session logical reads***: The sum of "db block gets" plus "consistent gets". This includes logical reads of database blocks from either the buffer cache or process private memory. The metric SESS_LOGICAL_READS_TXN measures this number of reads for each transaction, if this number can be high per second this improve from performance of system.

4. CONCLUSION

This paper has discussed several proposals for building balanced technique for concurrency control that enable concurrent access to data items with keeping consistency of database. Some proposals try to improve accuracy (correctness) of data, others are designed to reduce execution time and waiting time for read operations, finally some provide read and write facility based on assumption that conflict occurs rarely. As shown from this paper, lock based algorithms cannot provide Serializability, also they create deadlock but 2pl tries to ensure serializability but also it is not deadlock free, then this paper describe timestamp algorithm that ensure Serializability using the ordering of timestamps generated by the DBMS, they also solve problem of deadlock as there is no waiting for transaction, but it increases storage requirements (memory needs and database's processing overhead), then we described multi-version that also ensure serializability and avoid read rejection process as all read operation can be performed successfully on old versions of data, but it also may increase storage requirements to store number of versions, finally this paper describe optimistic technique that is based on assumption that conflict may occur rarely, also it can save some execution time than other techniques because lock can be done only at last phase(write), but it may aborts many transaction to enable committed one from completion of its process. We classify proposals into several categories based on the theory of concurrency control, we show that some proposals can differ widely but all discuss same subject.

5. REFERENCES

[1] Carlos Cornol and Steven Morris, "Database Systems, Design, Implementation and Management", United States of America,12th edition ,2016

[2] Samuel Kaspi and Sitalakshmi Venkatraman, "Performance Analysis of Concurrency Control Mechanisms for OLTP Databases", International Journal of Information and Education Technology, Vol. 4, No. 4, August 2014.

[3] Md. Anisur Rahman," An Efficient Concurrency Control Technique for Mobile Database Environment", Global Journal of Computer Science and Technology, Vol 13,No. 2,2013.

[4] D. Lomet et Al.," Multi-version Concurrency via Timestamp Range Conflict Management", IEEE 28th International Conference of Data Engineering (ICDE) ,1-5 April 2012

[5] Avi Silberschartz, Henry F.Korth and S.Sudarshan," Database System Concepts" ,NewYork McGraw-Hill, 1997, , 4th edition, pp591-638

[6] Quazi Mamuan and hidenor Nakazato, "timestamp based optimistic concurrency control", IEEE conference of TENCON 2005, Crown Promenade HotelMelbourne, Australia ,21 Nov - 24 Nov 2005.

[7] S. Shanwal and S. Kumar," Secure concurrency control algorithm for multilevel secure databases", The Next Generation Information Technology Summit (4th International Conference) , Noida,26-27 Sept. 2013

[8] S.Sippu and Soisalon-Soininen,"Transaction Processing, Data-Centric Systems and Applications", Springer international publishing Switzerland, Nov. 2014.

[9] Mohammad Sadoghi et Al," Reducing database locking contention through multi-version concurrency", Proceedings of the VLDB Endowment, Volume 7 Issue 13, August 2014

[10] Priyanka Kumar, Sathya Peri and K. Vidyasankar," A TimeStamp Based Multi-version STM Algorithm", Distributed Computing and Networking15th International Conference, Coimbatore, India, January 4-7, 2014

[11] Sonal Kanungo and Morena Rustom. D,"Analysis and Comparison of Concurrency Control Techniques",International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 3, March 2015.

[12] Wolf Stephan et al. " In Memory Data Management and Analysis" , Springer International Publishing Switzerland 2015 , August 26, 2013, pp.82-93.

[13] Dahlia Malkhi and Jean-Philippe Martin," Spanner's concurrency control",Newsletter ACM SIGACT News, Volume 44, Issue 3, September 2013 ,Pages 73-77.

[14] Ramez Elmasri and Shamkant B. Navathep," Fundamentals of Database Systems", Pearson,sixth edition,2011,pp 776-800.

[15] Rashmi Srinivasa, Craig Williams and Paul F. Reynolds Jr," A New Look at Timestamp Ordering Concurrency Control", database and expert Systems Applications, 12international conference, Munich, Germany, 3-5 sep. 2001.

[16] Jaypalsinh A. Gohil and Prashant M. Dolia, "Study and Comparative Analysis of Basic Pessimistic and Optimistic Concurrency Control Methods for Database Management System", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 1, January 2016.

[17] Ashish Srivastava, Udai Shankar and Sanjay Kumar Tiwari," A Protocol for Concurrency Control in Real-Time Replicated Databases System", International Journal of Computer Networks and Wireless Communications (IJCNWC), Vol.2, No.3, June 2012.

[18] Jose M. Faleiro and Daniel J. Abadi, "Rethinking serializable multiversion concurrency control", Proceedings of the VLDB Endowment, Vol. 8, No. 11, 2015.

[19] Per-Åke Larson et Al.,"High-performance concurrency control mechanisms for main-memory databases"," Proceedings of the VLDB Endowment", Volume 5, Issue 4, December 2011, PP. 298-309.

[20] NASER S. BARGHOUTI AND GAIL E. KAISER, "Concurrency Control in Advanced Database Applications", ACM Computing Surveys, Vol 23, No 3, September 1991.

[21] Peter Rob and Carlos Coronel," Database Systems: Design, Implementation, and Management", Joe Sabatino Publisher, tenth edition, 2013, USA. ISBN-13: 978-1111969608