Study of Locking Protocols in Database Management for Increasing Concurrency

Swati[1]
Research Scholar
Computer Science Engineering,
Amity University Haryana,
Gurugram, India
swattiguptta@gmail.com

Dr Shalini Bhaskar Bajaj[2]
Professor
Computer Science Engineering
Amity University Haryana,
Gurugram, India
sbbajaj@ggn.amity.edu.

Abstract: **Transaction performs lock and unlocking operations on the data items required in its execution. These operations on the data items are important in order to maintain consistency of the database as the data may be accessed by concurrently executing transactions. Designing locking and unlocking mechanism on the data items it involves fine tuning which involves the following factors a) level of granularity b) appropriate version to be locked c) supporting the compatibility mode. In this paper, two locking protocols are covered namely Multi version and Multi granularity locking protocol. The Multiple granularity locking protocol specify at which level locks can be applied, if the lock is applied on exact level of database, then definitely system performance can be improved and basically explore how can be increase the concurrency of MGL locking protocol by considering suitable example. An efficient locking technique is proposed by integrating multi version with multiple granularity in the hierarchical structure. This allows several requesting transactions to be executed in parallel by serving them an appropriate version to read while the write operation of some other transaction is in progress.**

Keywords: Concurrency, Atomicity, Multi Version, Granularity.

I. INTRODUCTION

In a database sharing environments manipulation of the data leverages on transaction processing. A transaction can be consider as a single atomic unit of the database whose processing results in a specific guarantee. These guarantees are specified in terms of different properties which are supported by the transactions, namely Atomicity, Consistency, Isolation and Durability (ACID properties) [1, 2]. Transaction process relies on the concept of concurrency control protocols (CCPs) which allows simultaneous execution of multiple transactions. A CCP lay down certain rules that govern the consistent state of the database by preserving desired properties of the transaction [1, 2]. Concurrency among transactions can be achieved through the locking mechanism. A lock is considered as a single atomic variable which has a data item associated with it [4]. Basically, it tells the feasibility of the operation. For the given data item a lock is applied to perform read (share lock) and write (exclusive lock) [12].There are several algorithms that support concurrency such as time stamping, multi-version, multiple granularity [1, 12].

In the literature there are multiple of locking algorithms that are used for synchronization in the database domains such as two phase locking, timestamp locking. In our paper, multi version and multiple granularity locking are being discussed which are used for carrying out locking mechanism.

In Multi Version Concurrency control (MVCC) each data item maintains multiple copies so as to increase the concurrency [4, 8, and 11]. At MVCC with each successful write operation a new version is produced .Transaction that perform read operation carries out writes on the same data item, as a result read is never obstructed by any update operation. In case of any read request it is being fulfilled by the previous version of the data until the write is being carried out and it has not committed [4, 5, and 12]. A variant of such a protocol is the Timestamp based Multi version [3]. In this variant, whenever a transaction *T* commits its newly created version is being read by another transaction that started after *T*. Upon receiving a write operation for a transaction *T* on the data item *d*, inspect if the version of *d* has already been read by some other transaction started after *T*, then in that case *T* is aborted. Otherwise, a new version of *d* is created. Finally, after all transactions which have produced versions read by *T* have been committed and all write operation are completed than the transaction *T* is finally committed to ensure serializability and recoverability [3, 5, and 10].

In Multiple Granularity Locking (MGL) [25] the resources to be locked is organized in the form of hierarchy as shown in fig 1.

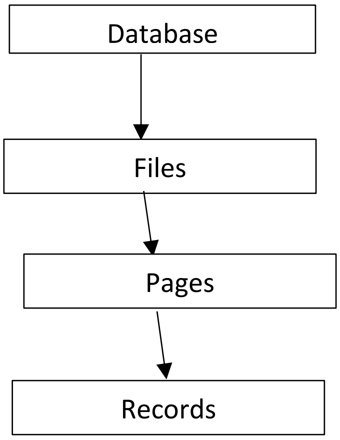

Figure 1: A Sample Locking Hierarchy [6]

Each node in the hierarchy represents the granularity at which lock can be applied. If a data item is required in exclusive mode, then the request is granted to the transaction by allowing the requestor to achieve an exclusive access to the data item while implicit lock is applied to all the ancestors of the node traversing along the path starting from the root node. [7, 13, 5].For each new request in non- exclusive mode the requestor achieves a shared lock on the particular data while all its ancestors are locked implicitly [6].In multiple granularity another locking mode namely intention mode is introduced to specify the fact that locking is done at the finer level of granularity [13, 14, 15]. The following locking modes exist in the multiple granularity locking protocol: Intension Share (IS), Intension Exclusive (IX), Shared (S), Exclusive (X), Share Intension exclusive (SIX).

In the paper, a new locking protocol is proposed in a hierarchical structure to carry out the locking operations among multiple transactions. Our novel contribution has revised the locking and version mechanism. Thereby, enabling synchronous locking operation among multiple transactions. The key insight behind proposing new concept is that it integrates the advantage of multi version and multiple granularity locking protocol.

II. RELATED WORK

The major research work carried out in the field of multi version and multiple granularity locking have being discussed.

a) Related work on Multi Version Locking

Wang et al. [3] proposed the multi version technique based on timestamp. It proposed the concept of decreasing the overload of any system by transforming the issues such as abort problem into the waiting state. This is achieved by maintaining a table that is being associated with the timestamp for each operation of the transactions. In case of any conflicts that arises during write operations the transaction which is older needs to commit first while keeping the younger transaction wait until the transaction which is older commits its operation.

Yingjun Wu et al [4] proposed the key version storage design in multi version database. The paper discussed three approaches to store the version 1) Append only storage: In this scheme the same storage space is being used to store the tuple versions that is being implemented either by using Oldest to Newest (O2N) approach. In the given O2N approach the head of the list points to the oldest version in the chain. It can further be implemented by new scheme that follows the Newest to Oldest (N2O) approach in which with each modification the head of the list.2) Time storage: In this scheme, a separate table is used to keep the older version of the data while maintaining the master copy in the main table. 3) Delta Storage: This scheme maintains a separate delta version sequence for updating the tuples in the DBMS.

Kaloian Manassiev et.al [8] proposed concurrency algorithm that deals with the concept of covering two things namely master update with scheduler support and update anywhere with no scheduler support. It covered the performance of the system by maintaining appropriate version. David Lomet et.al [12] proposed a conflict manager known as timestamp conflict manager (TCM) that associate any committed transaction with its transaction timestamp which is made dependable with transaction isolation order. In the TCM a series of timestamp is maintained for the transactions. The paper has formulated certain principles that control the access of transactional operations on the particular data. The timestamp range is adjusted whenever any conflict occurs so that read is ahead of write. This reduces transaction abort so that it there is no blocking of the operation in the case of contradictory operations. Per Ake Larson et.al [13] discussed a new locking concept that is being suitable for database system. It covers two main approaches to implement two multi version concurrency control (MVCC) methods, first covers optimistic using validation phase while the another one is based on pessimistic approach that practices locking technique.

Robert Gottstein et al [14] is based on the idea of index only visibility check that is being significantly used to decrease the amount of input-output storage accesses which thereby reduces the maintenance of index overhead.Hoda et.al [15] et al discussed that in order to determine the optimal version a minimum timestamp for each version can be maintained by a special variable that is stored in the linked list. Sadoghi et.al [9] suggested an index maintenance technique which is referred as indirection KV. This scheme helps in reducing the input output burden. It is achieved by retaining a single table that can contain both updated and historic data in a special table known as version enabled. Jose M et.al [16] suggested bohm which is a new concurrency protocol that is used in multi version system. In order to guarantee serializable execution it separates concurrency control and version management from transaction execution that ensures read is not blocked by any update operation. .

Jie Shao et.al [17] explains that in order to achieve read consistency the system support snapshot read which do not block write operation. They proposed an architecture which comprises of three main components: partition, DTM, consistency coordinator. The concept of Local transaction identifier (LTID) has been introduced to the number of snapshot for each version number. The table 1 shows the Comparative analysis of the work carried out recently in the field of multi version locking.

Table 1: Performance Metrics of Multi Version Locking

Reference No.	Proposed Technique	Contributions
[3]	Multi Version Protocol based on timestamp approach	It converts the rollback condition into waiting state thereby decreasing the abort rate.
[4]	Proposed new Multi version storage structure	Three storage schemes were proposed 1. Append storage 2. Time storage 3. Delta storage
[8]	Proposed Novel distributed Multi version concurrency scheme	For each update a new version is broadcasted at each site in the distributed environment
[12]	Proposed Timestamp based conflict manager scheme	The timestamp range is adjusted such that read is always ahead of write operation this in turn reduces the abort rate.
[13]	Designed Optimistic and Pessimistic Multi version scheme	It specifies which version is appropriate to read during each read request
[14]	Proposed Multi version index scheme	It reduces the maintenance of index overhead in multi version storage by providing index snippets
[15]	Proposed optimized garbage collection scheme	It discard the obsolete version by comparing the timestamp of each version with the current running timestamp.
[9]	Proposed KV indirection index scheme	It maintains single table for carrying current and historical data thereby reducing index overhead
[16]	Proposed Bohm new concurrency Multi version Scheme	It ensures read never blocks write operation.
[17]	Proposed architectural scheme prevalent in multi version system	It is based on the concept of local transaction identifier

b) Related Work on Multiple Granularity

Saurabh Kalikar et.al [18] present domlock which is a new multiple granularity locking protocol. The new locking protocol store the information of all the nodes by using a special concurrent pool data structure. The two steps are involved to lock any new node: (i) In order to check overlap between intervals the lookup pass is being used (ii) if no overlap is being identified by the lookup pass then a new entry is being inserted in order to apply a lock. Donatella Gubianieral [19] it represent the spatial data by providing multiple representations. It proposed a technique which provides an extension lead to the abstract model known as chrono geo graph. This allow the user to switch from one spatial entity to another by providing spatial aggregations.

The Liu et.al [20] presented a new multiple granularity system known as CTrace.The process traces is being explored by the user for providing the information regarding conceptual abstraction that provides granularity at different levels. In Jelena et.al [21] presented a Multi granularity middleware known as Generic Lock Service (GLS) that supports traditional lock interface. The functionality covers two main functions to perform the operation of acquiring and releasing a lock.

Table 2: Performance Metrics of Multiple Granularity Locking

Reference No.	Proposed Technique	Contributions
[18]	Proposed Domlock, a new multiple granularity technique	In order to maintain information about all the locking nodes a concurrent pool data structure is being used. This is helpful in identifying the overlap regions.
[19]	Proposed Chrono Geo Graph Multiple Granularity model	It represent a new spatial aggregation
[20]	Proposed CTrace a new multi granularity system	The process traces is being explored that specify conceptual abstraction in order to support granularity at different levels.
[21]	Generic Lock Service(GLS) a Multi granularity middleware	The two main functions namely lock declaration and lock initialization are used in this system.
[22]	New Multiple granularity Access schemes	It supports access at two different level namely instance access and class definition. In order to support concurrency the new schemes are highly useful.
[23]	Proposed Numlock a new multiple granularity technique	In order to provide optimal locking options it uses novel polytime option generation algorithm
[24]	HiFi, a new MGL locking protocol	In order to check the overlaps regions a novel indexing technique is used among multiple threads.

In Jun [22] presented a new locking scheme that deals in Object oriented database. It comprises of two types of access supporting at the instance and class level. In case of any conflicting operation a finer granularity is adopted that is useful in supporting both instance and class definition such that concurrency at higher level can be attained. The Kalikar et.al [23] proposed a Numlock, new locking technique which carries its works by choosing an appropriate locking combination which is optimum for any MGL requests that is being carried out by the transaction. Basically in this method in order to generate optimal locking option a new novel poly time generation algorithm is being used. Ganesh et.al [24] discussed Hifi which is a new locking protocol. It allows the combination of grained locks at finer granular with locking at hierarchical level. In order to perform the operation of two requesting threads it follows a novel indexing technique to check for the overlaps regions. The table 2 shows the recent development carried out in the field of multiple granularity.

III. PROPOSED LOCKING SCHEME IN HIERARCHICAL STRUCTURE.

The advantage of multi version is combined with multiple granularity that provides the following advantages:

a) Improving the existing compatibility matrix of multiple granularity.

b) At each granular level multiple versions is maintained so that read request by the transaction can be carried out with each write in progress by the another transaction

c) Enhanced concurrency and performance of the system.

Illustrative Example:

Let us consider the transactional operations to support our proposed approach which are discussed as follows:

Operation O1: It wish to perform read on node r1 in Shared mode and Operation O2 that wish to apply a write lock on the same node r1 as shown in fig. 2.

For performing this operation, hierarchically locking protocol MGL apply shared lock on node r1 and intension share lock to all its descendants' node till root node. The prevalent state of-the-art MGL locking approaches does not allow any other operation to perform write on node r1 while read is in progress. In such case the other operation O2 has to wait thereby decreasing the concurrency and came up with the improvement that allow both transactional operation O1 and O2 to proceed in parallel such that O1 can read the data item while O2 write is in progress. This is possible by keeping multiple version at each hierarchical level such that O1 finishes its work while write by O2 is in progress. The proposed scheme thereby increases concurrency and performance of the system

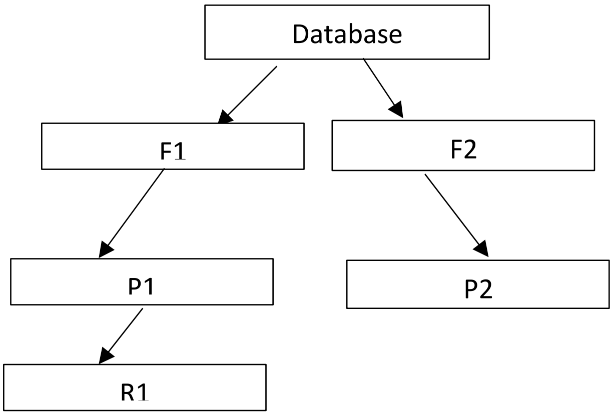

Fig 2: Locking request in Hierarchical Structure

IV. CONCLUSION

In a multi user distributed transaction the conflicting operations are common occurrence. These conflicting operations are synchronized in order to maintain the stable state of the database system. Locking is one of the most important paradigm that effects the system performance. In order to handle these situations there are several locking techniques proposed in the literature. Two locking techniques namely multi version and multiple granularity have been studied and came up with the improvement by proposing new concurrency control algorithm which is an amalgamation of both multi version and multiple granularity locking approach. It is suitable for database environment such that simultaneous execution of read and write operation can be carried out by the multiple transaction on the same data item in the hierarchical structure. Thus, our paper aims at improving the performance of the system by eliminating the waiting condition of read only transactions.

REFERENCES

[1] Bharat Bhargava "Concurrency Control in Database Systems" IEEE Transactions on knowledge and data engineering, VOL. 11, NO. 1, JANUARY/FEBRUARY 1999

[2] Qasim Abbas, Hammad Shafiq, Imran Ahmad, *|Mrs. Sridevi Tharanidharan" Concurrency Control in Distributed Database System" 2016 International Conference on Computer Communication and Informatics (ICCCI -2016), Jan. 07 – 09, 2016, Coimbatore, INDIA

[3] Wang Yujun, LI Junke The Solution to the Roll Back Problem in Multi version Concurrency Control Timestamp Protocol International Conference on Computer Science and Network Technology,pp 2803-2806,IEEE 2011

[4] Yingjun et.al "An empirical evaluation in memory multi version concurrency control" in VLDB vol 10, No.7, 2017.

[5] Wu, Y., Arulraj, J., Lin, J., Xian, R. and Pavlo, A., 2017. An empirical evaluation of in memory multiversion concurrency control. Proceedings of the VLDB Endowment, 10(7), pp.781-792

[6]Priyanka Kumar1,Sathya Peri1, and K. Vidyasankar2" A Timestamp Based Multi-version STM Algorithm" ICDCN , LNCS 8314, pp. 212–226, 2014.Springer-Verlag Berlin,Heidelberg 2014

[7] Justin Levandoski, David Lomet, Sudipta Sengupta, Ryan,Stutsman, and Rui Wang" Multi-Version Range Concurrency Control in Deuteronomy" 42nd International Conference on Very Large Data Bases, September 5th – September 9th 2016, New Delhi, India. Proceedings of the VLDB Endowment, Vol. 8, No. 13

[8] Kaloian, Manassiev, MadalinMihailescu, Cristiana Amza "Exploiting Distributed Version Concurrency in a Transactional Memory Cluster pp. 198- 208, New York, ACM 2006

[9] Mohammad Sadoghi1, Mustafa Canim1, Bishwaranjan Bhattacharjee1,Fabian Nagel3, Kenneth A. Ross1;" Reducing Database Locking Contention Through Multiversion Concurrency" 40th International Conference on Very Large Data Bases, September 1st 5 th 2014, Hangzhou, China. Proceedings of the VLDB Endowment, Vol. 7, No. 13

[10] Thomas Neumann Tobias Mühlbauer Alfons Kemper" Fast Serializable Multi-Version Concurrency Control for Main- Memory Database Systems" SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

[11] Rachael Harding Dana Van Aken" An Evaluation of Distributed Concurrency Control" Proceedings of the VLDB Endowment, Vol. 10, No. 5 Copyright 2017.

[12] David Lomet and Mohamed F. Mokbel" Locking Key Ranges with Unbundled Transaction Services" ACM. VLDB '09, August 24-28, 2009, Lyon, France.

[13] Per-Åke Larson1, Spyros Blanas2, Cristian Diaconu1, Craig Freedman1, Jignesh M. Patel2, Mike Zwilling1" High-Performance Concurrency Control Mechanisms for Main-Memory Databases"Published in the Proceedings of the VLDB Endowment, Volume 5 Issue 4,Pages 298-309,2011.

[14] Robert Gottstein, Ilia Petrov, Alejandro Buchmann "Append Storage in Multi-Version Databases on Flash "Published in Big Data. BNCOD, Lecture Notes in Computer Science, vol 7968, Springer, Berlin, Heidelberg,2013

[15]Hoda M. O. Mokhtar, Nariman Adel Hussein" A Novel Mechanism for Enhancing Software Transactional Memory"pp 278-283, July 07-09,Portugal, ACM, 2014.

[16] Jose M. Faleiro, Daniel J. Abadi "Rethinking serializable multiversion concurrency control" Published in the Proceedings of the VLDB Endowment, Volume 8 Issue 11, July 2015, Pages 1190-1201

[17]Jie Shao, Boxue Yin, Bujiao Chen, Guangshu Wang, Lin YangJianliang Yan, Jianying Wang, Weidong Liu "Read Consistency in Distributed Database Based on DMVCC" pp. 142-151, IEEE 23rd International Conference on High Performance Computing 2016.

[18] Saurabh Kalikar and Rupesh Nasre" Dom Lock: A New Multi-Granularity Locking Technique for Hierarchies" , ACM, PPoPP '16 March 12-16, 2016,Barcelona, Spain.

[19] Donatella Gubiani, Angelo Montanar "A conceptual spatial model supporting topologically-consistent multiple representations" ACM GIS '08, November 5–7, 2008.

[20] Qing Liu, Kerry Taylor, Xiang Zhao, Geoffrey Squire, Xuemin Lin, Corne Kloppers" CTrace: Semantic Comparison of Multi-Granularity Process Traces" SIGMOD'13, June 22–27, 2013, New York, USA.Copyright 2013.

[21] Jelena Anti´c, Georgios Chatzopoulos, Rachid Guerraoui, Vasileios Trigonakis" Locking Made Easy"ACM, ISBN 978-1-4503-4300-8/16/12. December 12 - 16, 2016, Trento, Italy.

[22] Woochun Jun" A multi-granularity locking-based concurrency control in object oriented database systems "Published in Journal of System and Software in Elsevier,Volume 54 ,issue ,November,2000, Pages 201- 217

[23] Kalikar, S. and Nasre, R NumLock: Towards Optimal Multi-Granularity Locking in Hierarchies, In Proceedings of the 47th International Conference on Parallel Processing, August 13–16, 2018, USA.

[24] Ganesh,Saurabh Kalikar, Rupesh Nasre "Multi-granularity Locking in Hierarchies with Synergistic Hierarchical and Fine-Grained Locks" Springer International Publishing AG, part of Springer Nature , LNCS 11014, pp. 546–559, 2018.

[25] J. N. Gray, R. A. Lorie, and G. R. Putzolu. Granularity of locks in a shared data base. In Proceedings of the 1st International Conference on Very Large Data Bases, VLDB '75, pages428–451, New York, NY, USA, 1975. ACM. ISBN 978-1-4503- 3920-9. doi: 10.1145/1282480.1282513.