

# IMPROVING THE EFFICIENCY OF SNAPSHOT ISOLATION

R.Suresh<sup>1</sup>, V.Amirtha Maria Cecilia<sup>2</sup>, G. Icewerya<sup>3</sup>, R. Thilagavathi<sup>4</sup>

Assistant professor, Department of Information Technology, Sri Manakula Vinayagar Engineering College,  
Puducherry, India<sup>1</sup>

UG student, Department of Information Technology, Sri Manakula Vinayagar Engineering College, Puducherry,  
India<sup>2,3,4</sup>

**Abstract:** As transaction processing became an integral part in real time applications, isolation concepts are used to maintain its integrity. The weaker isolation levels aimed at higher concurrency but are prone to integrity problems such as lost updates, phantom reads. To avoid the above problems, SI which is a multi versioning concurrency control is adapted. Therefore in this paper, we compare the performance of snapshot isolation with traditional isolation levels and techniques like transaction chopping and escrow locking are implemented to improve the efficiency of snapshot isolation.

**Keywords:** Concurrency control, Serializability, Snapshot Isolation(SI), (S2PL) Strict two Phase Locking

## I. INTRODUCTION

Concurrent transaction processing is essential for efficient performance of database systems. Real time database applications receive hundreds of requests per second and average response time has to be minimized without compromising on correctness of execution<sup>[2][1]</sup>.

The correctness is ensured by Serializable execution while response time is determined by extent of concurrency allowed by the system<sup>[4]</sup>. Hence, serializability and optimum concurrency are the most desirable characteristics of a concurrency control mechanism<sup>[2]</sup>. Strict two-phase locking (S2PL) ensures Serializable execution but suffers from limited concurrency. Many systems that implement weaker levels of serializability, allow non-serializable schedules, causing inconsistencies in the database<sup>[2]</sup>. Thus there is a trade-off between level of isolation and degree of concurrency. Snapshot Isolation, a multi-version concurrency control, avoids most of the typical anomalies and yet provides better concurrency than S2PL<sup>[3][4]</sup>.

SI has been supported by some of the leading database systems such as Oracle, Microsoft SQL Server, and PostgreSQL etc. However, not all schedules under SI are Serializable. So, a given set of transactions can be allowed to execute at a lower isolation level only if it is safe to do so<sup>[3]</sup>. Therefore the serializable snapshot isolation is implemented to ensure serializability in concurrent transactions avoiding the classic anomalies of snapshot isolation such as read anomaly and write anomaly.

The paper is organized as follows. Section 2 briefs the prior research. Section 3 discusses about the performance of the isolation levels. The techniques of transaction chopping and escrow locking are discussed in Section 4 for testing serializability.

## II. PRIOR RESEARCH

Isolation is the property of distributed systems that ensures that concurrent transactions do not interfere with each other. The generalized definitions of the isolations are

proposed in<sup>[1]</sup> specifies a common definition for not only locking but also optimistic and multi versioning concurrency control. The paper [2] defines the phenomena in which the weaker isolations fail in providing the data integrity such as non-repeatable reads, phantom reads and dirty reads. This paper introduces the newly defined isolation called Snapshot isolation, which is based on multi versioning concurrency control overcomes all the classic integrity problems stated above. The paper [3] describes the classic problem of read only anomaly in snapshot isolation. It states that the snapshot isolation, though avoids most of the concurrency problems, show some of its the anomaly behavior in certain areas such as lost update, write skew and read only anomaly when the serializability is not provided. The paper [4][5] proposes the concept of making snapshot isolation Serializable. This paper states the mechanism in which the snapshot isolation executes serially offering good performance while reducing the common serializable violations through careful analysis of the transaction programs and is implemented in the snapshot databases such as Berkeley Db etc.

## III. RESEARCH PROPOSAL

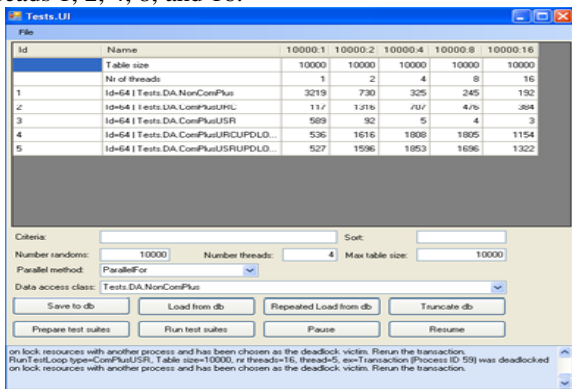
In the paper, there is comparison between the performance between Snapshot isolation with the traditional isolation levels.. Traditional lock based concurrency system reduces the performance because of lock contentions and waiting time<sup>[4]</sup>. Thus Snapshot isolation which is a multi version concurrency control is used to avoid the classic problems of lower level isolations. SI produces non-serializable schedules causing anomalies such as Read skew, Write skew and lost updates<sup>[3]</sup>. Thus the SI is made serializable in order to avoid the serializability issues exhibited by SI. <sup>[5]</sup>The Serializable Snapshot isolation allows to implement multiple concurrent transactions to be executed serially based in the "First Committer Wins" rule. It prevents the execution of the concurrent transaction without any Read-

Write anomalies<sup>[4]</sup>. A log file is created for each access to the database which stores both the time of modification and updates made to the data<sup>[3]</sup> Multiple versions are created for each time of execution and are stored in the form of log files.<sup>[4]</sup> The time taken for executing multiple transactions is greatly reduced when compared to executing under single transaction.

#### A. Performance of isolation levels

The basic part in the database applications is to read one record from the database, alter some fields on the record, and save the updated record back to the database and it is usually done in one transaction.<sup>[1]</sup> In this, the functionality given by enterprise services (com+) is used to implement the transaction. The performance of the isolation levels are measured by running this transaction repeatedly and concurrently by different threads. The combination of these isolation levels with the sql update lock is used of the throughput manipulation. A transaction with read committed isolation level may read one record and proceed with the other and the record that was read is not locked, and a second transaction may update the record. With transaction isolation level serializable, the record that was read is locked until the end of the transaction, and no transaction may update that record before the first transaction has completed. Therefore, with the read committed isolation level, fewer transactions may run simultaneously. However, because the read committed isolation level does not lock the records when reading from the database, two simultaneous transactions may read the same record at the same time. The first transaction which will update the record will succeed, but the second will fail, a concurrency violation error will be thrown. With the serializable isolation level, the same sequence will result in a deadlock.

Sql server provides the update lock, which instructs the database engine to lock the record until the end of the transaction. This will enter as many records as specified in the number random textbox with a random generated name. The number of successful calls will be displayed in the grid. The test suite, starts with one thread in the first column, and continues with powers of two up to the power given in the text box number threads. For example if number threads is 4, the test suite will run test cases for threads 1, 2, 4, 8, and 16.



Id	Name	10000-1	10000-2	10000-4	10000-8	10000-16
	Table size	10000	10000	10000	10000	10000
	Nr of threads	1	2	4	8	16
1	Id=64   Tests.DA.NonComPlus	3219	730	325	245	192
2	Id=64   Tests.DA.ComPlusURC	117	1216	167	476	384
3	Id=64   Tests.DA.ComPlusUSR	589	92	5	4	3
4	Id=64   Tests.DA.ComPlusURCUPDLO...	536	1616	1809	1805	1154
5	Id=64   Tests.DA.ComPlusUSRUPDLO...	527	1536	1953	1696	1322

Figure: 3(a)Threads are been executing concurrently generated by the isolation levels using com+ and update lock.

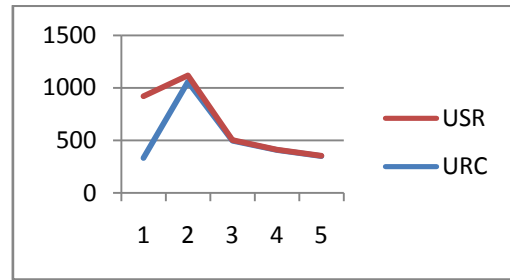


Figure3(b) Read committed and serializable isolation level constraints without update lock

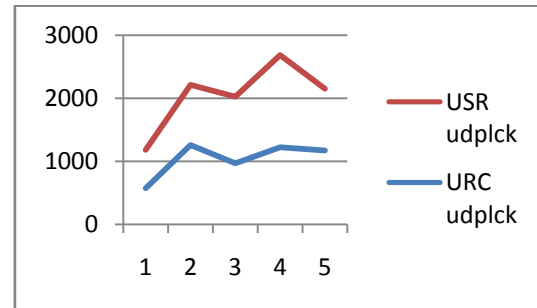


Figure 3(c) Read committed and serializable isolation levels with update lock.

The concurrent threads that are been executing in the com+ and non-com+ classes with and without updlock have been produced random values which indicates that read committed and serializable isolation have similar values in case of single thread execution but when multiple threads are been executed it is shown figure:3(b)(c) that serializable isolation level is effective with and without updlock.

#### IV. TRANSACTION CHOPPING

Long-lived transactions often reduce the system throughput significantly by denying other short transactions executing concurrently, access to certain data items results in long waits or abortion of other transactions.<sup>[7]</sup> In such cases, the long-transactions are chopped into smaller pieces which can be interleaved with other transactions allowing greater degree of concurrency<sup>[8]</sup>.

When chopping the transactions, there is a precaution to avoid chopping that might lead to non-serializable or non-recoverable schedules.<sup>[7]</sup> In many real-world applications, logical transactions spread across multiple-screens are implemented as a series of database transactions.<sup>[8]</sup> When logical transactions are split into multiple database transactions, the programmer pays little attention to the issue of concurrency control and serializability of the logical transactions.<sup>[7]</sup> Thus, it is necessary to have appropriate guidelines for chopping<sup>[8]</sup>. Therefore the chopping must undergo four dependencies based on the occurrence of the transaction.

#### A. Transaction Dependencies

The transaction chopping mechanism has variant four ways of dependency which is been categorized based on the Shasha et al<sup>[6]</sup> concept of splitting the conflicts of transactions where it is based on the occurrence of the

read–write transactions. There is variant difference in the commit and the abort transactions when there is concurrent execution of the transaction which is proceeding<sup>[9]</sup> during the banking segments like the banking operations , risk management , asset management management .The dependencies of the transaction chopping mechanism are –

- Read –Write dependency
- Write –Read dependency
- Write-Write dependency

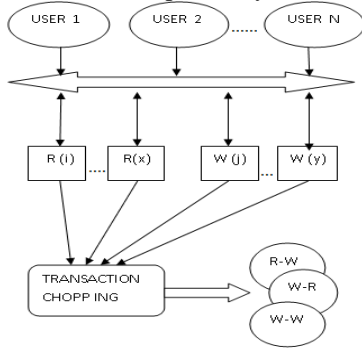


Figure4(a )User driven transactions by the chopped dependency.

### I) Read-write dependency

With the phenomenon of the snapshot isolation , there is a concept that the read transaction never block the write transaction , so the version read by the transaction in the log file will not have the effect on the read transaction <sup>[8]</sup>.

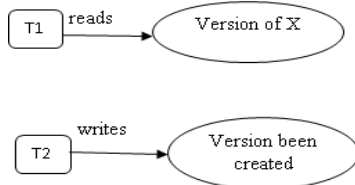


Figure4(b) T1, T2 performs read-write dependency.

For example the transaction T1 will read the record and version get updated and then write transaction will be updated in the database.

### II) Write-read dependency

In the optimistic concurrency control , the transactions are independent and separated . If the transaction T1 writes the data item X it gets updated in the log file and when the another transaction T2 reads the same data item it recovers the data from the last record of the updated version so it maintains the stability<sup>[5]</sup> and overcomes the lost update anomaly which improves interoperability .

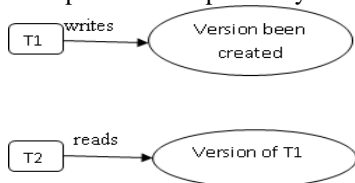


Figure4(c) T1,T2 performs write-read dependency

### III) Write-write dependency

If transaction T1 update a data item X while a concurrent transaction T2 also tries to update the same data item , it

will cause one of the transactions to abort, so the first update will not be lost According to First-Committer-Wins rule,<sup>[2]</sup> it is not possible to have two concurrent transactions which both commit and both modify the same data item<sup>[5]</sup>.

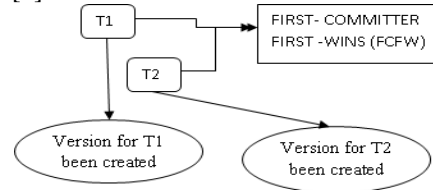


Figure4(d) T1,T2 performs write-write dependency by using FCFW rule.

### B. Chopping of the transactions

The transaction chopping mechanism is been evaluated with the execution of single query and multiple query processing. In practice, implementations of SI usually prevent a transaction from modifying an item if a concurrent transaction has already modified it as. It has higher consistency than repeatable read and overhead of maintaining row versions.

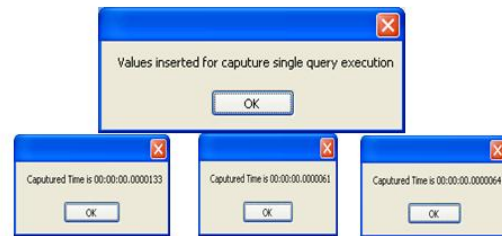


Figure 4(e) The capture time during the single query execution when values are inserted.

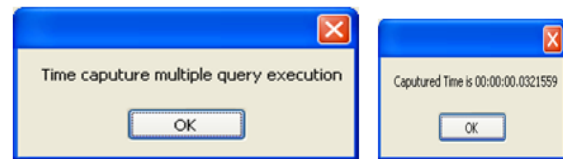


Figure 4(f) The capture time during the multiple query execution when values are inserted.

With the comparison of the execution of the transaction chopping mechanism with single query been executed one by one the captured time is less to the captured time of the multiple query executed without the transaction chopping .mechanism .thus the efficiency of chopping mechanism of the transactions prove to be effective in case of the concurrent execution of the transaction in the snapshot isolation.

### C. Escrow Locking

Escrow locks provide a method for permitting long lived transactions to update frequently accessed records without forbidding other users from accessing them<sup>[8]</sup>. However, it is most relevant and applicable to quantities that can be changed incrementally.

The basic idea underlying Escrow transactions is as follows:

- a. They involve certain fields and quantities and a class of tests and update operations on these quantities<sup>[8]</sup><sup>[6]</sup>.
- b. Whenever a transaction makes an attempt to perform Escrow-type-update, if the operation is approved then the system must guarantee that the update can be performed at any time, and in any order with any subset of updates<sup>[7]</sup>.
- c. If there is a test clause is a necessary condition for the update, then the guarantee should hold for the validity of test clause . Any update added to the set of already applied updates should not make the test condition for some other previous update invalid.
- d. For each request to update a field, a record is created which contains the relevant information such as transaction ID, parameters of request and field being updated<sup>[8]</sup> For each field, a range of values which can be assumed is maintained.
- e. When a transaction that made a request to update a certain field eventually commits/aborts, Thus, all updates are actually applied only at the commit. Thus it can be seen that the purpose behind designing Escrow locks is to improve upon concurrency in long-lived transactions which take very long time to complete and hence taking the possession of the lock throughout the execution of transaction could severely hamper concurrency<sup>[9]</sup> Thus, escrow locking and chopping could be considered as alternatives for improving concurrency. Having escrow locking and chopping together could further increase the throughput.

#### D. System Architecture

The user starts the transaction and the log file is been created to store the modifications made on the data and is versioned for future reference. The user gives the long lived transactions as an input to the system, where the transactions are chopped and given as output to the user<sup>[7]</sup>. Once the chopped transactions get executed, the user transaction are been performed and the changes is tracked and versions are saved in the database .

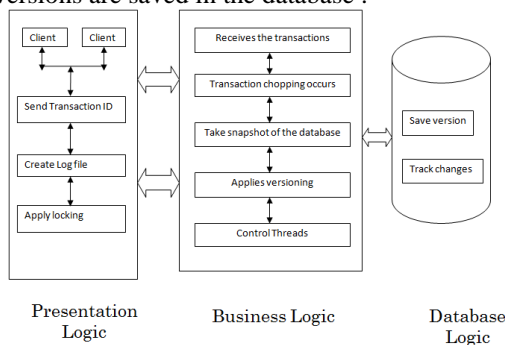


Figure 4(e) System architecture

#### V. CONCLUSION AND FUTURE WORK

Thus we conclude the paper that the Serializable implementation of Snapshot isolation provides a better performance in response time since the well known anomalies such as read only anomaly, write skew and lost update have been overcome because of the serial execution of the transactions by the transaction chopping

and escrow locking techniques. The future work of this paper is to implement and test snapshot isolation in multi core servers for the synchronous execution.

#### REFERENCE

- [1] A. Adya, B. Liskov, and P. E.O'Neil"Generalized Isolation Level Definitions". In ICDE 2000.
- [2] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P.O'Neil, "A Critique of ANSI SQL Isolation Levels," Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data, pp. 1-10, May 1995.
- [3] Alan Fekete." Allocating isolation levels to transactions. "In PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 206-215, New York, NY, USA, 2005.ACM Press.
- [4] S. Jorwekar, A. Fekete, K. Ramamritham, and S. Sudarshan, "Automating the detection of snapshot isolation anomalies," in Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB2007),Vienna, Austria, 2007
- [5] Alan Fekete, Elizabeth O'Neil, and Patrick O'Neil. A read-only transactions anomaly under snapshot isolation. SIGMOD Rec., 33(3):12 - 14, 2011.
- [6] A. Fekete, "Serializability and snapshot isolation," in Proceedings of the 10th Australasian Database Conference (ADC '99), Auckland, NewZealand,1999.
- [7] Alan Fekete and Denis Shasha. "Making Snapshot Isolation Serializable." ACM Transactions on Database Systems, 37(2):492-528,2010.
- [8] Michael J. Cahill, Uwe Rohm and Alan D. Fekete." Serializable Isolation for Snapshot Databases". Page 4, 2012.
- [9] Dennis Shasha, Fran\_cois Llirbat, Eric Simon, and Patrick Valduriez. "Transaction chopping: Algorithms and performance studies". ACM Transactions on Database Systems, 20(3):325 - 363, 2011.
- [10] Patrick E. O'Neil. "The escrow transactional method" ACM Trans. Database Syst., 11(4):405 - 430, 1999.
- [11] R. Schenkel and G. Weikum"Integrating Snapshot Isolation into Transactional Federation". In CoopIS 2002.
- [12] R. Schenkel, G. Weikum, N. Weienberg, and X. Wu. "Federated transaction management with snapshot isolation" In 8th Int. Workshop on Foundations of Models and Languages for Data and Objects -Transactions and Database Dynamics, 1999.
- [13] M. Alomari, M. Cahill, A. Fekete, and U. Rohm. "The cost of serializability on platforms that use snapshot isolation". ICDE 2008.
- [14] Bernstein, Lewi.P, and Lu. S. 2000," Semantic conditions for correctness at different isolation Levels". In Proceedings of IEEE International Conference on Data Engineering (Feb.). IEEE Computer Society Press, Los Alamitos, Calif., 57-66.
- [15] Jung, H., Han, J.H., Fekete, A., R'ohm, U., Yeom, H.Y " Performance of serializable snapshot isolation on multicore servers"(2013).
- [16] Shiyong Lu , Patrick O'Neil , "Serializable Snapshot Isolation for Replicated Databases in High Update Scenarios",2012