Tradeoff in Optimistic Concurrency Control Algorithms for Centralized Database Systems

***Dr. Kamal M. Solaiman*[1]*,Tarik Idbeaa*[1]*, Dr. Tariq Khalifa*[1]*, Dr. Ayad A. Keshlaf*[2]**

[1]*School of Science , Al Jabal Al Gharbi University*

[2] *Industrial Research Center, Tripoli*

Abstract:

Optimistic concurrency control is widely studied in the literature due to the properties of non-blocking and deadlock free execution especially in the domain of real-time systems. In this paper we review the substantial research of optimistic concurrency control protocols. We characterize them into four categories and explore their properties. Then we investigate the general concepts and properties related to Optimistic concurrency control. Finally, we demonstrate a comparison table between the varieties of these protocols.

***Keywords**: Optimistic concurrency control; real-time system; Serializability; Rollback; Starvation*

I. Introduction

Concurrency control is a mechanism for coordinating access to shared data in order to prevent any unexpected results and maintain consistency. Two important concurrency control approaches have been investigated in the literature: Pessimistic Concurrency Control(PCC) and Optimistic Concurrency Control(OCC). PCC is based on mutual exclusion. Shared data locked by only one process to prevent other processes from accessing it. When this process finishes execution, all data locked by this process will be released[1]. OCC reduces locking overhead by allowing multiple uncontrolled reads to share the data. New updates are thereby checked to prevent conflicts between them; if new updates violate state consistency, these updates will be canceled. If new updates maintain consistency, then these updates can be copied to the original database [2]. OCC was an attractive solution because of the properties of non-blocking and deadlock free execution (especially in real-time systems). Performance evolution studies of OCC techniques can be found in [3-8].

Serializability property is maintained to ensure database consistency[2, 9]. The serializability means that there is at least one serial schedule that leads to the same final state of database[10-12]. The rest of the paper is organized as following: Section 2 classifies the main approaches of OCCPs. Section 3 introduces variety of OCC concepts and properties.

Section 4 presents the comparison criteria tables. Discussion is presented in section 5. And finally the conclusion and future work presented in section 6.

II. Optimistic concurrency control protocols (OCCPs):

OCCPs are designed to achieve a reduction of locking overhead. They rely explicitly on the assumption that conflicts between concurrent transactions infrequently occur. The earliest optimistic concurrency control protocol has been introduced by Kung and Robinson[2]. In this protocol transaction execution time is divided into three phases (read phase, validation phase and write phase). During the read phase, transactions access data without any restrictions, making copies of original data in their private workspace (read set). In the validation phase, after a transaction has read all data and all computation has been done, a resolution policy has to be applied to ensure serializability. If no conflicts between concurrently running transactions have been detected, then the transaction progresses to the write phase to update the original data (write set) and commit. Otherwise, the transaction aborts. However, the write phase can be eliminated in the case of read only transactions (query).

In this section, OCCPs are classified into four categories: OCCPs based in 2PL certification, OCCPs based on serialization graph, OCCPs based on timestamp and integrated OCCPs. These categories are described briefly below:

2.1 OCCPs based in Two Phase Lock (2PL) certification:

In 2PL certification protocols scheduler maintains read set for every concurrently running transaction contains all entries read by such transactions. And write set for every concurrently running transaction contains new updates intended to be made by such transactions. When a transaction Ti reaches the end of its read phase and scheduler receives a request for commit Ti. The scheduler validates transaction Ti by looking at the intersections between read sets RS (Ti) and write sets WS (Ti) of the validating transaction Ti and all other concurrently running transactions Tj. The schedule checks every concurrently running transaction Tj to determine if RS(Ti) ∩WS(Tj), WS(Ti) ∩ RS(Tj) or WS(Ti) ∩ WS(Tj) $\neq \emptyset$. If so, then this transaction has conflict and resolution policy need to be applied. Otherwise transaction commits and removed from the set of active transaction. [13]

In SGT a scheduler maintains a serialization graph of the history representing the execution controls. During the execution, the scheduler maintains the Serialization Graph (SG) by adding edges between concurrent transactions nodes corresponding to all reads and writes operation requested without consideration of SG being acyclic. When a transaction Ti finishes execution and scheduler receives request for commit Ti, it checks if Ti lies on acyclic of the SG. If so, then this indicates that there has been a conflict operation inserted to the schedule and some resolution policy needs to be applied to resolve this conflict. Otherwise, there is no conflict operation. SGT scheduler provides some flexibility but maintaining SG overhead and checking for cycles adds extra cost to this technique [13-15].

2.3 OCCP Based on Timestamp:

In timestamp based OCCPs, a timestamp (TS) is associated to every data item and every running transaction. This timestamp is used to ensure serializability order after transaction has executed and ready to commit. This simply achieved by using Timestamp rule, which defined as following:

In the execution if some operation Oi belong to Ti precedes some conflicted operation Oj belongs to Tj then TS(Ti) < TS(Tj), therefore, any other conflicted operations belong to Tj not allowed to precede any conflicted operations belong to Ti. [16-18].

Some OCCPs are designed to use timestamp intervals [19-22], This timestamp interval is associated to every transaction, and will dynamically adjust whenever data items are accessed. If a conflict has occurred, the timestamp interval will be shutdown. OCCPs based on timestamps generally show high degree of concurrency, guarantee the deadlock free property, and provide relatively a smaller number of unnecessary rollback overhead. In contrast, timestamp based OCCPs drawback is the large overhead of maintaining timestamp management [21].

2.4 Integrated OCCP:

Integrated OCCPs provide both OCC and locking techniques. This combination was formed in order to manage aborting and blocking in a more effective manner. The first hybrid approach was introduced in [23] which proposed using OCC for first run and then if the transaction is rolled back, automatically change the type to locking by inserting a lock before each access to data item. This approach provides an advantage for long lived transactions which are more likely to conflict and roll back with short transaction, and in some cases may lead to starvation. Varieties of hybrid protocols have been investigated in the literature and can be viewed in [6, 24-29].

III. Concepts/Approaches Descriptions:

In this section we explore eighteen concepts and approaches that should be taken into an account when designing OCCPs. These aspects and approaches includes conflict detection, conflict resolution, starvation problem, number of rollbacks, unnecessarily rollback, partial rollback, transaction length, query consideration, transaction arrival rate, correctness criteria, transaction granularity, static/dynamic schemes, silent/broadcast commit, rerun policy, speculative CC, parallel validation, priority inversion problem and deadline-cognizant.

3.1 Conflict Detection:

In OCC, conflicts are detected after granule access. Where checking for serializability is done later at the validation phase. Deferent mechanisms can be used for conflict detection proposes, such as backward oriented optimistic concurrency control (BOCC) , forward oriented optimistic concurrency control (FOCC), timestamps, and serialization graphs schemes [9, 30].

- BOCC: In this scheme, intersection between the read set of a validating transaction T and the write sets of all other concurrently running transactions that have finished their read phase before T have to be checked. If there is an intersection, aborting T is the only way to resolve this conflict. [9]

- FOCC: In this scheme, intersection between the write set of a validating transaction T and the read sets of all other concurrently running transactions that have not yet finished their read phase, have to be checked. If there is an intersection, one of the following resolution strategies can be used: 1.Delay T and restart the validation phase later. 2. Abort any transaction has a conflict with T and commits T. 3. Abort T. [9].

- Timestamp scheme: a timestamp is assigned for every granular and transaction. In every access to granule, the transaction's timestamp is checked against timestamp of the last transaction that has accessed this granule in order to satisfy timestamp order rule [16, 30].

- Serialization graph testing scheme: The concurrency control manager maintains a serialization graph representing the execution ordering of all transactions in the history. If a conflict occurs then a cycle will be produced by serialization graph[13, 30].

3.2 Conflict Resolution:

Conflicts between transactions can be divided into two types: Reconcilably Conflicting Transactions and Irreconcilably Conflicting Transactions [31].

- Reconcilably Conflicting Transactions are transactions that have only read-write conflicts with validating transaction; these conflicts can serialized without any abortion.

- Irreconcilably Conflicting Transactions are transactions that have both read-write and write-write conflicts with validating transaction. When these conflicts occurred, then transactions are involved in a nonserializable execution. Restarting either a validating or running transaction(s) involved in this conflict is required. In this case, some consideration has to be taken regarding to transaction priority, length, deadline and the amount of transaction execution has already done and will be wasted if a transaction aborted [21, 22, 31, 32].

3.3 Starvation Problem:

Starvation occurs when a transaction is continually rolled back due to conflicts. Starvation is more likely to occur for long-live transactions and for that access to same granule often. Starvation problem simply could be resolved by giving priority to a starved transaction or blocking the whole database to give chance for a starved transaction to commit [2]. Many solutions in [6] [33-37] have been investigated in order to solve the starvation problem.

3.4 Number of Rollbacks:

Restarting conflicted transactions may directly increase the probability of having the same conflict again. So, by waiting some period time before next restart may help to decrease the number of rollbacks. However, delaying transactions, especially in real time systems, may cause failure of meeting transactions deadline. By allowing transaction to restart until they successfully commit may increase the probability of starvation problem occurrence, especially for long transactions. [2, 16, 38]

3.5 Unnecessarily Rollback:

Conflicts between running transactions can be divided into two kinds of conflicts; serious conflicts and non-serious conflicts.

- serious conflict is a conflict leads to unexpected results in database state and conflict resolution has to be taken against this conflict to preserve database consistency and integrity[33].

- Non-serious conflict is a conflict does not lead to database inconsistency state and there is no need to restart conflicted transactions or run conflict resolution scheme [21, 31, 33, 39, 40].

3.6 Partial Rollback:

Partial rollback is a technique to reduce wasted execution caused by transactions rollbacks. This technique is implies rolling back only the conflicted part of the transaction. Which consequently reduces the cost and the time of transaction execution especially in the long transactions [41].

3.7 Transaction Length:

Long transactions have higher risk to starve than short transactions due to two reasons:

- Long transaction needs longer execution time which increases the chance of affection by other committed transactions.

- Long transaction accesses larger number of elements which increases probability of confliction on these elements with other transactions. [6, 33] Giving similar chance of committing to both short and long transactions is an important aspect which has to be taken into consideration.

3.8 Query consideration:

Query transactions (read only transactions) have no write phase and have no computation overhead. Thus, giving some flexibility in the validation phase can give great impact especially for query application. Protocols in [2, 9, 33] have been designed to give special treatment to query transactions in order to increase the performance.

3.9 Transaction Arrival Rate

When number of running transactions accesses the same data elements allowed growing without restrictions. Then the number of conflicts between these running transactions grows as will. This in sequence increases percentage of transactions rollbacks. Therefore, Limiting the number of running transaction that accessing the same data elements plays an important role of reducing rollback overhead [17].

3.10 Correctness Criteria :

Serializability is the basic fundamental approach for correctness criteria in most OCCPs [2, 9, 13, 17, 21, 41-43]. Serializability means that there is at least one serial schedule leads to the same final state of database [13]. However, in some circumstances weakening isolation level can have great impact to increase transactions throughput especially in a long transactions and read only transactions [33].

3.11 Transaction Granularity:

Transactions in OCCPs backup data items in its private workspace. This is an extra consumption of the main memory space and the size of data is considered as granule (Word, Page, or Object) It is an important issue in designing OCCPs especially in case of insufficient memory [44].

3.12 Static/Dynamic data access Schemes :

Reading data from database to transactions private workspace can be performed by two schemes: static access and dynamic access.

- Static data access scheme: all data elements will be read in the beginning of transaction execution. This basically gives more flexibility of designing validation mechanism. However, this helps to increase contention in the system because data held for longer time [13, 14, 45-47].

- Dynamic data access scheme: Data elements are read one by one as they are needed. Although, this scheme gives more complication of designing validation mechanism, dynamic access scheme reduces data contention compared to static access scheme because data held for shorter time [45-47].

3.13 Silent/Broadcast Commit :

When transactions successfully finish validation and write phases, transactions commits by one of two commitment schemes: silent commit and broadcast commit.

- Silent commit scheme: In this scheme, a transaction becomes aware of conflicts only at validation time. The running transactions continue execution till the end of their read phase and enter the validation phase [45-47].

- Broadcast commit scheme: Committed transaction advertises its commit to all conflicted transaction in order to restart these conflicted transactions as soon as possible. This technique avoids wasted execution done by conflicted transactions and unnecessarily waiting. Broadcast commit has an advantage in comparison to previous silent commit by providing early conflict detection [45-47].

3.14 Rerun Policy :

A rerun policy is a concurrency control technique based on virtual run [48, 49] and aims to reduce I/O restart overhead . In this technique a transaction is allowed to continue execution even if it has conflicted and remarked to restart. The reason is that giving this transaction chance to prefetch all needed data to its private workspace in the memory. This transaction then restarts as soon as it finishes first execution. In the second run (rerun) there is no need to read again from data storage, instead, the transaction reuse the prefetched data stored in the memory from when it first read. Access invariant property has to be guaranteed when using this approach. Access invariant property means that any two executions of the same transaction must always access the same data items, even if these executions are separated by other conflicted transactions[48, 49].

4.15 Speculative Concurrency Control (CC):

Speculative CC technique uses redundant transactions to start as early as possible on an alternative schedule when a conflict is detected. This redundant transaction is called a transaction shadow. If conflict in the original transaction is resolved and successfully commits, then this transaction's shadow must be aborted. On the other hand, if the original transaction fails to commit, then this transaction's shadow is adopted, instead from restarting original conflicted transactions from scratch, this techniques offers better opportunity for real-time transactions to commit within their deadline expiry. However, this advancement costs extra memory and processing resources when transactions succeed to commit and the other running shadow are discarded. [42, 50-58]

3.16 parallel Validation :

For implementation simplicity; transactions in the validation and write phases executes in critical section. This particularly reduces the parallelism [2, 6, 9, 21]. Parallelism can be increased by allowing more than one transaction validating and committing. However, eliminates critical section in the validation and write phase adds complexity to OCC technique [2, 6, 33].

3.17 Priority Inversion Problem:

Transactions processing in real-time systems are priority restricted and criticalness. Problem of priority inversion is occurs when a higher priority transaction has to wait for execution of lower priority transaction which has already started. This waiting may cause lose of higher priority transaction deadline. In designing real-time OCCPs, some consideration need to be paid to resolve such kind of problems [31].

3.18 Deadline-cognizant :

Timeliness is the primary performance measure in real-time OCCPs, not the response time and throughput. Scheduling of concurrent transactions based on priority consideration to minimize the number of missed deadline transactions rather than fairness. There are many deadline-cognizant studied in the literature [31, 59-64], in the following brief description of four well known policies.

- OPT sacrifice policy: Used in OCCPs when validation transaction restarts if one or more conflicting transactions have higher priority than the validating transaction [7, 65, 66].

- No Sacrifice policy: in this policy transaction is grantee to commit if it started the validation phase and all other irreconcilably conflicted transaction have to be restarted as soon as conflicted has been detected[31].

- Wait-50 policy: wait-50 is compromising the two previous policies (OPT sacrifice, No Sacrifice). Validating transaction in wait-50 policy is delayed if more than 50% of conflicted transactions have higher priority than the validating transaction. Otherwise it proceed execution to the write phase [7, 65].

- Feasible sacrifices: Feasible sacrifices implies that validating transaction which has a conflict with higher priority transaction will not be restarted unless this validating transaction still has enough time to meet its deadline [31]. This technique saves resources and execution time.

IV. Comparison Criteria:

Salient important eleven concepts from previous section have been compared in table 1. These aspects include (Conflict resolution, dynamic access schemes, Number of rollback, unnecessarily rollback, Partial rollback, transaction length, , parallel validation, Starvation resolution, Query consideration, broadcast commit and rerun policy.

Table 1. Shows the comparison of 2PL certification Optimistic Concurrency Control Protocols

Category	NO	Year	Reference(s)	Criteria / Approach	Conflict resolution	Dynamic access	NO. of rollback	Unnecessarily rollback	Partial rollback	Transaction length	Parallel Validation	Starvation resolution	Query consideration	Broadcast commit	Rerun Policy
2PL certification OCCPs	1	1981	Kung H.T., Robinson J., T. [2]	Serial Validation	abort validating transaction	×	✓	×	×	×	×	✓	✓	×	×
				Parallel validation	abort validating transaction	×	✓	×	×	×	✓	✓	✓	×	×
	2	1984	Theo H. [9]	BOOCC	(commit or abort) validating transaction	✓	*	×	×	×	×	*	✓	×	×
				FOOCC	(delay or abort) validating transaction or abort conflicted transactions	✓	*	×	×	×	×	*	×	×	×
	3	1991	Jiandong H.,John A., Krithi R.,Don T. [6, 37]	OCCL-SVW	1- Abort conflicted transactions. 2- Abort validating transaction if its priority < all conflicted Ts. 3- Delay validating transaction if its priority not the highest among conflicted transactions.	✓	×	×	×	✓	×	✓	×	×	×
				OCCL-PVW		✓	×	×	×	✓	✓	✓	×	×	×
	4	1991	Yu S. P., Dias M. D..[25, 67]	OCC with broadcast during rerun	Rerun nonfirst run conflicted transactions	✓	×	*	×	×	×	×	×	✓	✓
	5	1992	O'Neil E. P., Ramamritham K., Pu C. [38]	Predictable transactions execution	If planning execution after prefetch phase could be constructed within the deadline then transaction grant. Otherwise transaction aborts.	✓	✓	✓	✓	×	×	✓	×	×	✓
	6	1994	Rainer U. [33]	EOT marker	abort validating transaction	*	*	✓	×	*	×	*	×	×	×
				Snapshot validation with critical section	abort validating or conflicted transaction	✓	*	✓	×	✓	×	✓	✓	×	×

				Snapshot validation without critical section	abort conflicted transaction	✓	*	✓	×	×	✓	×	×	✓	×
	7	1995	Thomasian A. [41]	Checkpointing	Abort conflicted transaction	*	×	*	✓	×	×	*	×	✓	×
	8	2004	Wang Y., et al [68]	DAEO	Abort conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	9	2011	Kamal S. Graham M. [69]	Later validation/ earlear write	Abort conflicted transactions	✓	×	×	×	×	×	×	×	×	✓

× - Aspect does not apply. ✓- Aspect is applied. * - Aspect did not mention.

Table 1. (continues) shows the comparison of Serialization Graph and Timestamp Based Optimistic Concurrency Control Protocols

Category	NO	Year	Reference(s)	Criteria / Approach	Conflict resolution	Dynamic access	NO. Of rollback	Unnecessarily rollback	Partial rollback	transaction length	Parallel Validation	Starvation resolution	Query consideration	Broadcast commit	Rerun Policy
OCCPs based on SG	1	1987	Philip B., Vassos H., Nathan G. [13]	Basic SGT	Abort newly arrival conflicted transactions	*	×	×	×	×	×	×	×	×	×
	2	1989	Marzullo K., [15]	Priority SGT	Abort less priority conflicted transactions	*	×	×	×	×	×	×	×	×	×
	3	2000	Victor L., K.-W. L.,[14]	Conflict free scheduling	Delay newly arrived conflicting transactions until running conflated transactions commit	×	No rollback			×	×	✓	×	×	×
based on Tim	1	1987	Ryu I., Thomasian A. . [45]		abort validating transaction	✓	×	×	×	*	×	×	×	✓	×

	2	1993	Lee J., Son H. S.. [21] [31]	OCC-TI	restart validating or conflicted transactions	✓	×	✓	×	×	×	×	×	×	×
	4	1995	Kwok-Wa L., Kam-yiu L., Sheung-lun H. [17, 70]	OCC-DA	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	5	1997	Konana P., Lee J., Ram S. [43]	Revised OCC-TI	restart validating or conflicted transactions	✓	×	✓	×	×	×	×	×	×	×
	6	1999	Lindström J., Raatikainen K. [22, 40]	OCC-DATI	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	7	1999	Juhnyoung L. [39]	Precise serialization.	Abort conflicted transactions	*	*	✓	×	×	×	×	×	×	×
	8	2000	Lindström J. [19, 40]	Revised OCC-TI	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	9	2000	Lindström J.,Raatikainen K [32, 40, 71]	RTDATI, PDATI	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	10	2002	Lindström J. [20] [40]	OCC-IDATI	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×
	11	2004	Wang Y., et al. [68]	OCC-CS	Restarting conflicted transactions	*	*	✓	×	×	×	*	×	×	×
	12	2005	Qilong H., Zhongxiao H.[72]	MVOCC-TFD	restart validating or conflicted transactions	×	×	✓	×	×	×	×	×	×	×
	13	2005	Mamun Q. E. K. ,Nakazato H. [16]	TS based OCC	restart validated transaction	✓	×	✓	×	×	×	×	×	×	×
	14	2008	Bai T., Liu Y., Hu Y.[73]	OCC-TSV	restart validating or conflicted transactions	*	×	✓	×	×	×	×	×	×	×

× - Aspect does not apply. ✓- Aspect is applied. * - Aspect did not mention.

V. Analysis and Discussion:

This section presents analyses of the resulted data presented in the previous section. It clearly shows that there is a considerable research that effort have been done in OCC algorithms. Table 1 illustrates that weakness aspects in some algorithms can be strong aspect in others and vice versa, also there is a similarity in some other Aspects. But there is no one optimal algorithm that has all criteria supported [74].

From table 2 we identified that unnecessarily rollback criterion has got the highest percentage of supported factor which is about 58.6%. However, partial rollback criteria have got the lowest percentage which is about 6.9 %. Although, number of rollback, transaction length, and parallel validation, query consideration, broadcast commit and rerun policy criteria have got low percentages (10.3 %, 10.0 %, 10.0 %, 13.3 %,13.3 %, 10.0 %) respectively. The starvation resolution criteria were a little bit higher about (23.3 %).

Table 2

factors / Criteria	Supported	Not supported	Not mentioned	Percentage of supported
Dynamic access	13	4	13	43.3 %
NO. Of rollback	3	19	7	10.3 %
Unnecessarily rollback	17	10	2	58.6 %
Partial rollback	2	27	0	6.9 %
transaction length	3	25	2	10.0 %
parallel validation	3	27	0	10.0 %
Starvation resolution	7	18	5	23.3 %
Query consideration	4	27	0	13.3 %
Broadcast commit	4	26	0	13.3 %
Rerun policy	3	27	0	10.0 %
SUM	55	210	29	18.7 %

There are many of algorithms conducted in the study (13 out of 30) did not mention clearly what kind of data access schemes Static or Dynamic are based on. However, 43.3 % of the algorithms conducted on the survey were using dynamic access scheme.

From table 1 we clearly identified that although; conflicts were resolved with deferent techniques in included algorithms. Aborting validating transaction or conflicting transactions are the most used schemes.

From this analysis, we can identify that there were focus in reduction of unnecessarily rollback resulted from non-serious conflicts, which greatly benefit from timestamp techniques. On the other hand, there was not enough concern about reducing the overhead caused by multiple necessarily rollback resulted from serious conflicts.

Criteria: number of rollback, transaction length, and parallel validation, query consideration, broadcast commit and rerun policy got less attention in the literature. And more work is really needed to be done on order to add more advancement in OCC.

VI. Conclusion And future work

In this paper we have reviewed OCC techniques studded in literature in order to identify the strengths and weaknesses aspects between them and explore their general properties. From this revision we have concluded the following:

The main shortcoming of OCCPs is the both necessarily and unnecessarily rollback overhead which is an expensive cost of the system resources and time. The survey shows that extensive research has been made for the sake of reduction of the unnecessarily rollback overhead gained from timestamp ability to distinguish between serious and non-serious conflicts. However, existence concurrency controls algorithms still suffer a weakness in reducing multiple necessarily rollback overhead which also may become expensive if the system faces high transactions contention level.

Another drawback is the static OCC overhead resulted from the techniques adopted in the existing OCCPs. This basically wastes a certain percentage of the total execution time in the system regardless to the contention changes. Designing a dynamic OCC that uses changeable OCC overhead depends on the level of transactions contention is still a great challenge.

Our future work :

In our future work we are designing an OCC algorithm capable of adjusting concurrency control overhead in run-time execution, with careful consideration to the level of transactions contention and overhead caused by multiple necessarily rollbacks.

References:

1. *Eswaran, K.P., Gray, J. N. , Lorie, R. A. , Traiger, I. L. , The notions of consistency and predicate locks in a database system. Commun. ACM, 1976. 19(11): p. 624-633.*
2. *Kung, H.T., Robinson, J. T. , On optimistic methods for concurrency control. ACM Transactions on database Systems, 1981. 6,2:p.213-226.*
3. *G., R.A.H., Scheduling real-time transactions: a performance evaluation, in ACM Transactions on Database Systems (TODS). 1992. p. 513-560.*

4. *Robert A., H.G. Scheduling real-time transactions: A performance evaluation. in In Proceedings of the 14th International Conference on Very Large Data Bases 1988. Los Angeles, Ca,.*
5. *Huang J., S.A.J., Towsley D., Ramamritham K., , Experimental evaluation of real-time transaction processing. In Proceedings of the lOth Real-Time Systems Symposium, 1989.*
6. *Jiandong H., J.S., Krithi R., and Don T.,, Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes. Proc. 17th Conf. Very Large Databases, 1991: p. 35-46.*
7. *Jayant H., M.C.M.L., Data access scheduling in Firm Real-time database systems. Real-Time Systems Journal, 1992. 4: p. 203-241.*
8. *Jiandong H., J.S., Krithi R. , Priority Inheritance in Soft Real-Time Databases. Journal of Real-Time Systems, 1992. 4(3): p. 243-268.*
9. *Theo, H., opservation on optimistic concurrency control systems. Inform. Systems, 1984. 9: p. 111-120.*
10. *Bernstein, P.A., V. Hadzilacos, and N. Goodman, , Concurrency Control and Recovery in Database Systems. 1987: Addison- Wesley, Reading.*
11. *Papadimitriou, C.H., The serializability of concurrent database updates, in ACM Trans. Database Syst. 1979. p. 631-653.*
12. *Garcia-Molina H., U.D.J., Widom J. , Database Systems: The Complete Book. 2001: Prentice-Hall.*
13. *Philip B., V.H., Nathan G., Concurrency Control and Recovery in Database Systems. 1987: Addison- Wesley, Reading.*
14. *L., V.L.K.-W., Conflict free transaction scheduling using serialization graph for real-time databases, in Syst. Softw. 2000. p. 57-65.*

15. *Marzullo K., Concurrency control for transactions with priorities. 1989, Department of Computer Science, Cornell University, Ithaca.*
16. *Mamun Q. E. K. , N.H., Timestamp based optimistic concurrency control. in TENCON2005, 2005: p. 1-5.*
17. *Kwok-Wa L., K.-y.L., Sheung-lun H. , Real-time optimistic concurrency control protocol with dynamic adjustment of serialization order, in on Proc of IEEE Real-Time Technology and Application Syniposium,. 1995. p. 174-179.*
18. *Victor C. S. L., K.-w.L., Sheung-lun H., , Concurrency control for mixed transactions in real-time databases. IEEE Transactions on Computers, 2002. 51(7): p. 821-834.*
19. *J., L., Extensions to optimistic concurrency control with time intervals, in In Proceedings of 7th International Conference on Real-Time Computing Systems and Applications. 2000, IEEE Computer Society Press: Cheju Island, South Korea. p. 108-115.*
20. *J., L., Integrated and adaptive optimistic concurrency control method for real-time databases, in International Conference on Real-Time Computing Systems and Application. 2002.*
21. *Lee J., S.H.S., Using dynamic adjustment of serialization order for real-time database systems. Proc. 14th IEEE Real-Time Systems 1993: p. 66 - 75.*
22. *Lindström J., R.K. Dynamic adjustment of serialization order using timestamp intervals in real-time databases. in In Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications,IEEE Computer Society Press. 1999.*

23. *Georg, L. Concurrency control in data base systems: A step towards the integration of optimistic methods and locking. in In Proceedings of ACM '82. ACM. 1982. New York.*
24. *Haran B., I.H., Towards a self-adapting centralized concurrency control algorithm, in n SZGMOD 84. ACM, New York. 1984. p. 18-31.*
25. *Yu S. P., D.M.D., Analysis of hybrid concurrency control schemes for a high data contention environment, in in IEEE Trans. Software Eng. 1992. p. 118-129.*
26. *Lam K., S.H.S., Hung S., A priority ceiling protocol with dynamic adjustment of serialization order, in In 13th IEEE Conf. on Data Engineering (ICDE'97). 1997: Birmingham.*
27. *Graham P., B.K. Effective Optimistic Concurrency Control in Multiversion Object Bases. in In Proceedings of International Symposium on Object Oriented Methodologies and Systems (ISOOMS). 1994.*
28. *Sang H. Son, J.L.Y.L., Hybrid Protocols Using Dynamic Adjustment of Serialization Order for Real-Time Concurrency Control. Real-Time Systems Journal, 1992. 4: p. 269-276.*
29. *Sang S., J.L., A new approach to real-time transaction scheduling, in Proceedings of Fourth Euromicro workshop on Real-Time Systems. 1992: Athens, Greece. p. 177-182.*
30. *Yu S. P., W.K., Lin K., SON H. S. , On Real-Time Databases: Concurrency Control and Scheduling, in In Proceedings of IEEE. 1994. p. 140-57.*
31. *J., L., Concurrency Control Algorithms for Real-Time Database Systems. PhD Dissertation January 1994, University of Virginia.*

32. *Lindstrom J., R.K., Using Importance of Transactions and Optimistic Concurrency Control in Firm Real-Time Databases, in Proc. 7th International Conference on Real-Time Systems and Applications (RTCSA '00). 2000: Cheju Island, South Korea. p. 12-14.*
33. *U., R., Optimistic Concurrency Control Revisited. Arbeitsbericht Institut f ¨ur Wirtschaftsinformatik der Westf¨alischen WilhelmsUniversit¨at M¨unster, 1994.*
34. *Peinl, P., Reuter, A. , Empirical comparison of database concurrency control schemes, in Proceedings of the 9th Znternutionul Conference on Very Large Data Bases. 1983: Florence. p. 97-108.*
35. *Pradel, U., Schlageter, G. , Unland, R. Redesign of optimistic methods: Improving performance and availability. in In Proceedings of the 2nd International Conference on Data Engineering IEEE Computer Society Press. 1986.*
36. *Thomasian., E.R.a.A. A New Distributed Optimistic Concurrency Control Method and a Comparison of its Performance with Two-Phase Locking. in In Proceedings of Tenth ICDCS. 1990.*
37. *Jiandong H., J.S., Concurrency Control in Real-Time Database Systems: Optimistic Scheme vs. Two-Phase Locking, in A Technical Report , COINS 90-66. 1990, University of Massachusetts.*
38. *Patrick O., K.R.C.P., Towards predictable transaction executions in real-time database systems. 1992: Amherst.*
39. *L., J., Precise serialization for optimistic concurrency control. Elsevier Science B.V. The Netherlands, 1999. 29(2): p. 163-179.*
40. *J., L., Optimistic Concurrency Control Methods for Real-Time Database Systems, S.o.P.A.R. A-2003-1, Editor. 2001, University of Helsinki Department of Computer Science.*

41. *Thomasian A., Checkpointing for optimistic concurrency control methods. Knowledge and Data Engineering, IEEE Transactions on, 1995. 7(2): p. 332-339.*
42. *B., A., Speculative Concurrency Control: A position statement. Technical Report TR-92-016. 1992, Computer Science Department, Boston University, Boston.*
43. *Konana P., L.J., Ram S., Updating timestamp interval for dynamic adjustment of serialization order in optimistic concurrency control-time interval (OCCTI) protocol. Elsevier Science B.V. , 1997: p. 189-193.*
44. *Harris T., L.J., Rajwar R., Transactional Memory, ed. M.D. Hill. 2007.*
45. *Ryu I., T.A., Performance analysis of centralized databases with optimistic concurrency control. Elsevier Science Publishers B. V., 1987. Volume 7(Issue 3).*
46. *Alexander, T. Analysis of some optimistic concurrency control schemes based on certification. in In Proceedings of the 1985 SIGMETRICS Conference on Measurement and Modeling of Computer Systems. 1985.*
47. *Theodore, J., Analysis of Optimistic Concurrency Control Revisited. 1992.*
48. *Franaszek P., R.J., Thomasian A. , Concurrency control for high contention environments. ACM Trans. Database Syst. 17 No.2, 1992: p. 304-345.*
49. *Peter F., J.R., Alexander T. Franaszek A. P., Robinson T. J., Thomasian A. , Access invariance and its use in high contention environments, in Proc. 6th Intl. Conf on Data Engineering. 1990: Los Angeles. p. 47-55.*

50. *Bestavros A., B.S., SCC-nS: a Family of Speculative Concurrency Control Algorithms for Real-Time Databases, in Proc. Third Int'l Workshop Responsive Computer Systems. 1993.*
51. *B., A., Speculative Concurrency Control, in Technical Report TR-93-002. 1993, Computer Science Department, Boston University, Boston.*
52. *Bestavros A., B.S., Time liness via speculation for real-time databasees in In Proceedings of RTSS'94: The 14th IEE Real-Time System Symposium. 1994: San Juan , Puerto Rico.*
53. *Bestavros A., B.S., Value-cognizant speculative concurrency control, in In Proceed- ings of VLDB ' 9 5 : The International Conference on Very Large Databases. 1995: Switzerland.*
54. *Bestavros A., B.S., Value-cognizant speculative concurrency control for real-time databases. Information Systems, 21(1):75-101 (1996). , 1996. 21: p. 75-101.*
55. *Jun C. Yan_Li Z., Y.S.G., A New Speculative Concurrency Control Protocol and The Analysis base on Petri Net, in Computer Engineering and Applications. 2009. p. 121-123.*
56. *Juna C., Y.f.W., Jian_ping W., Concurrency Control Protocol for Real-Time Database and The Analysis Based on Petri Net. Advanced Materials Research 2011. 143-144: p. 12-17.*
57. *Azer B., S.B., Euthimios P., Performance Evaluation of Two-Shadow Speculative Concurrency Control. 1993, Boston University.*
58. *Haubert, J., B. Sadeg, and L. Amanton. Improving the SCC protocol for real-time transaction concurrency control. in Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on. 2003.*

59. *Datta A., S.H.S., Kumar V., Limitations of priority cognizance in conflict resolution for firm real-time database systems. IEEE Transaction on Computers 49 2000: p. 483-501.*
60. *C., J.H.M.L.M., Earliest deadline scheduling for real-time database systems, in Proc. 12th Real-Time System Symp. 1991.*
61. *Liu, C., L ., Layland, J., W . , Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment. In Journal of ACM, 1973. 20: p. 46-61.*
62. *L., J.H.M.C.M., Value-based scheduling in real-time database systems, in VLDB J. 1993.*
63. *Jayant H., M.C., Miron L., On Being Optimistic about Real-Time Constraints, in Proceedings of the 1990 ACM SIGACT-SIGART-SIGMOD Symposium on Principles of Database Systems (PODS). 1990. p. 331-343.*
64. *Robert A., H.G., Scheduling real-time transactions: a performance evaluation. ACM Transactions on Database Systems (TODS), 1992. 17(3): p. 513-560.*
65. *Jayant H., M.C.M.L., Dynamic Real-Time Optimistic Concurrency Control, in Proceedings of the 11th IEEE Real-Time Systems. 1990: Orlando, Florida. p. 94-103.*
66. *Lee J., S.H.S., Performance of concurrency control algorithms for real-time database systems, in Performance of concurrency control mechanisms in centralized database systems, K. Vijay, Editor. 1995, Prentice-Hall, Inc. p. 429-460.*
67. *Philip Y., D.D., Performance analysis of optimistic concurrency control schemes with different rerun policies, in in Proceedings of the Fifteenth Annual International. 1991: Japan. p. 294 - 300.*

68. *Wang Y., W.Q., Wang H., Dai G. Dynamic adjustment of execution order in real-time database. in In Proceedings of 18th International Parallel and Distributed Processing Symposium. 2004.*
69. *Kamal S., G.M., Later Validation/Earlier Write: Concurrency Control for Resources-Constrained Systems with Real-Time properties, in 30th International Symposium on Reliable Distributed Systems. 2011: Madrid.*
70. *Lam K., L.K., Hung S., Optimistic Concurrency Control Protocol for Real-time Databases. Elsevier Science Inc., 1997. 38: p. 119-131.*
71. *Lindström J., R.K., Using real-time serializability and optimitic concurrency control in firm real-time database. , in in proceedings of the 4th IEEE International Baltic Workshop on DB and IS Baltic DB IS'2000. 2000. p. 25-37.*
72. *Qilong H., Z.H., Real-time Optimistic Concurrency Control based on Transaction Finish Degree. Journal of Comuter Science, 2005: p. 471-476.*
73. *Bai T., L.Y., Hu Y., Timestamp vector based optimistic concurrency control protocol for real-time databases, in in Wireless Communications, Networking and Mobile Computing 2008. WiCOM '08. 4th International Conference. 2008. p. 1-4.*
74. *C., M., Less Optimism About Optimistic Concurrency Control, in In Proceedings of the Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing. 1992. p. 199-204.*