Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 4, No. 2, May 2013), Pages: 63-73
www.jacr.iausari.ac.ir

A New Concurrency Control algorithm in Temporal Database

Mirsaeid Hosseini Shirvani[1]✉, Mehran Mohsenzadeh[2], Seyed Majid Hosseini Shirvani[3]

1) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
2) Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
3) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
mirsaeid_hosseini@iausari.ac.ir; mohsenzadeh@srbiau.ac.ir; majid_hosseini_111@yahoo.com

Received: 2013/01/18; Accepted: 2013/02/25

Abstract

The large number of applications manages time varying data. Existing database technology seldom supports temporal database, TDB, according to time aspects. These intrinsic temporal database applications rely on such database which stores and retrieves time referenced data. Moreover, applications need to be managed on common data items access simultaneously and to be precluded from inconsistency as soon as possible which is the main task of concurrency controller or CC in short. The method used by CC in typical DB differs from its attitude with TDB. The variety algorithms were proposed regarding to TDB properties by reduction of granule size and decreasing the rate of conflicts to satisfy good performance, but none of them has achieved robust results. There are two categories of CC such as pessimistic and optimistic. In this paper new approach, with considering the TDB aspects, based on optimistic method has been suggested. It reclines the size of granule as data item appropriately and recognizes the conflicts swiftly. Consequently, we compare our proposed algorithm with pervasive 2PL-pessimistic approach. The outcome shows that new proposed algorithm has high degree of trade off with satisfying near conflict time detection and high rate of parallelism metrics.

Keywords: Temporal Database, Concurrency controller, 2PL-pessimistic, time varying data

1. Introduction

The main goal of DBMS is to execute different transactions in which Atomicity, Consistency, Isolation and Durability properties are preserved. For the sake of conciseness we use ACID. Concurrency controller component or CC in short is one of the most important elements of DBMS that its task is to execute all transactions, which potentially have conflicts, according to serialisable theory that is the result of execution is equivalent to serial execution of all transactions[1]. Moreover, for each two transactions working on same data item simultaneously in which at least one of them intends to write on determined data item conflict happens. Since variety execution of transactions' operations has different results therefore CC component must interleave operations and execute them in order to achieve parallelism and to reach the same result as if transactions would be executed in serial. The object of interleaving is to run the operations in parallel which do not have any conflict. In this way we can enhance the

rate of parallelism. In order to manage data which are associated to time and time history for retroactive TDB is widely being used[2,3,4,5,6]. The large numbers of database applications are temporal in nature such as financial applications like portfolio management, banking, accounting, personnel record keeping and scheduling applications like hotel reservation, airlines, trains and so on[7]. In the aforementioned circumstances applications rely on such database which stores and retrieves time referenced data. Concurrency controller's task is so sophisticated. There are two categories for CC such as optimistic and pessimistic[8]. In optimistic approach, requested resources are simply dedicated to transactions, before commit operation, database soundness will be audited. On the other hand, allocating the resources will be probed against requesting resources in pessimistic. Indeed, it will be investigated whether in this moment is there any conflict or not. This work happens periodically during time slice. Each approach has some merits and demerits. In optimistic method with reduction of granule size, not only we can decline the rate of conflicts, but also we can ameliorate the degree of parallelism[8]. Although its weakness is high rate of abortion and long time that resources take to be locked during validation phase for the sake of prevention of the other transaction access. On the other hand, pessimistic method can recognize conflicts as soon as possible, but the sole shortcoming is low degree of parallelism[9]. The aim of this paper is to represent a new method based on optimistic approach in order to brisk conflict detection and enhancing the degree of parallelism with appropriate decreasing of granule size. In this way we apply serializable theory as strong formal verification tool[10]. The rest of this paper is organized as follows. In section two, temporal database semantics is described. Afterward in section three, new suggested algorithm is brought in details. Consequently, noticeable results are elaborated in section four and remarkable sum up is taken into account in conclusion section.

2. Temporal DB semantics

The main core of TDB concentrates on data independent modeling which associates time and facts. Database which models and stores the portion of real world is so-called modeled entities or micro real world[11]. Real world's aspects are represented with different structures named by database entities. The term "fact" is used for each logical statement that can be assigned by true or false. There are other features related to facts like valid time and transaction time[12,13]. Valid time covers past, present and future[11]. It is the time that the fact is true in real world. All facts have valid time and it is usually defined by users and for the sake of some reasons it may not necessarily be recorded in DB. The next is transaction time which the fact is current in database. Despite valid time, transaction time of each database entity related to objects and values is not necessarily logical statement. Transaction time is duration from insert to delete. Consequently, this aspect makes the deletion to be completely logical rather physical. It means physical deletion is not applied and the record stays in DB, but its belonging to current state is ceased. Transaction time provides time varying state of DB whereas keeping track and responsibility are important. Whenever data are facts transaction time seems to be redundant and in this case valid time and transaction time are the same. Anyway, we use both transaction time and valid time as bitemporal aspect[11]. Unlike valid time, transaction time is determined by DBMS and includes begin of creation through end of current time. Time has different aspects like current time that we

nominate it "now"[14]. The time "now" is unique and it does not have reuse property. According to TDB features, usage of relational model for modeling, storing and reporting is a daunting task[15]. In TDB another model is applied. Reference in [16] implies several models. For instance, consider a video store that hires out video CDs and customers have unique CustomerID and rented CD has TapeNum as unique key. The aim is to store and retrieve the CDs rented during May 2007. On second of May customer C101 rents CD T1234 for three days long. This CD returns on 5th. Also, on 5th customer C102 rents CD T1245 for open ended return, but it returns on 8th. On 9th, customer C102 rents CD T1234 to be returned on 12th. On 10th rental CD is extended to 13th, but it does not return until the 16th. Video store keeps the records of rental DCs in a typical relation named by CheckedOut. Figure1 illustrates scenarios regarding to Bitemporal Conceptual Design Modeling that for abbreviating we nominate it BCDM[16].

CustomerID	TapeNum	T
C101	T1234	{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4), . . . , (UC, 2), (UC, 3)(UC, 4)}
C102	T1245	{(5, 5), (6, 5), (6, 6), (7, 5), (7, 6), (7, 7), (8, 5), (8, 6), (8, 7), . . . , (UC, 5), (UC, 6), (UC, 7)}
C102	T1234	{(9, 9), (9, 10), (9, 11), (10, 9), (10, 10), (10, 11), (10, 12), (10, 13), . . . , (13, 9), (13, 10), (13, 11), (13, 12), (13, 13), (14, 9), . . . , (14, 14), (15, 9), . . . , (15, 15), (16, 9), . . . , (16, 15), . . . , (UC, 9), . . . , (UC, 15)}

Figure1. Bitemporal Conceptual CheckedOut

This model stamps tuples in the form of (*tt, vt*) which is correspondent to facts and attribute *T* is used for it. Timestamps as ordered pairs (*tt, vt*) in *T* means that at the current time *tt* represented fact is validated at time *vt*[17]. Special value "Until Changed", *UC*, is the symbol that includes the related facts which remain as a portion of DB in current state and appear in ordered pair of each time. Figure2 depicts three timestamps graphically which are bitemporal elements.

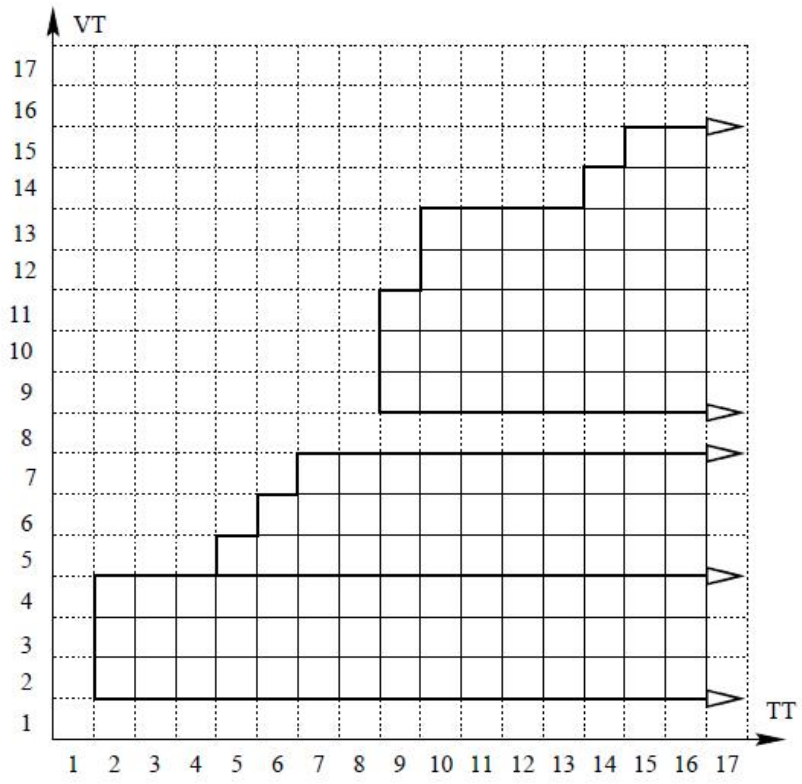

Figure2. Time stamps value according to scenarios

In general, time domain is continuous, but in two dimensional space with transaction time and valid time. Arrow direction toward right is used for *UC*. Although the relation seen in figure1, which is in BCDM model, is in 1NF (First Normal Form) level, but if the length and volume of tuples alternate then its management is impractical. There arebetter representations for this goal. Figure3 presents diverse data modeling for same temporal information.

CustomerID	TapeNum	T_s	T_e	V_s	V_e
C101	T1234	2	*UC*	2	4
C102	T1245	5	7	5	*now*
C102	T1245	8	*UC*	5	7
C102	T1234	9	9	9	11
C102	T1234	10	13	9	13
C102	T1234	14	15	9	*now*
C102	T1234	16	*UC*	9	15

(a)

CustomerID		TapeNum	
[2,*Now*] × [2,4]	C101	[2,*Now*] × [2,4]	T1234
[5,7] × [5,∞]	C102	[5,7] × [5,∞]	T1245
[8,*Now*] × [5,7]		[8,*Now*] × [5,7]	
[9,9] × [9,11]		[9,9] × [9,11]	T1234
[10,13] × [9,13]		[10,13] × [9,13]	
[14,15] × [9,∞]		[14,15] × [9,∞]	
[16,*Now*] × [9,15]		[16,*Now*] × [9,15]	

(b)

Figure3. Two different data modeling for CheckedOut

In Figure3.a, it have used fix length for tuples whereas attributes T_s,T_e,V_s and V_e are transaction start time, transaction end time, start of valid time and end of valid time respectively[18]. In Figure3.b, the relation does not belong to 1NF level and has low performance. In spite of BCDM model where relation must be updated for each clock tick, relations remain updated in two last models with using the term "now". After database introduction, we should specify concepts and operations in TDB. Concepts include Generic Key (GK), every combinatorial of attributes, is unique without temporal considerations[2]. For instance, in bitemporal relation in figure3.a for rental videos the GK is combinatorial of CustomerID and TapeNum. The next concept is General Tuple (GT). GT is every set of tuples with the same GK. In Temporal relation CheckedOut of figure3.a the fourth tuple through the end pertain to the same GT. Operations are Read, Insert, Delete and Update that are defined below:

Read(*R, Sel_cond[, ts[, te]]*)

Insert(*R, gk, vts[, vte][, attribute-name:data-value]*)

Delete(*R, Sel_cond[, ts[, te]]*)

Update(*R, Sel_cond[, ts , te] , attribute-name: data-value[,attribute-name: data-value]*)

In Read operation transaction T_i requests to read from tuples of relation R that satisfysel_condcondition during the interval [ts,te]. The concurrency controller receives "Readmessage(T_i, Rel, gk, vts, vte)" for each selected valid tuple that has gk as its GK. Moreover, the period [vts, vte] is the time overlap between read operation and valid time of specified tuple. Insert operation details are the same as Read operation. In Delete operation transaction T_i requests to delete tuples from relation R whereas sel_cond is satisfied and valid time belongs to interval [ts, te]. Concurrency controller

receives "Deletemessage(T_i, Rel, gk, vts, vte)" for every deleted generic tuple. Update operation details are the same as delete operation.

3. Our New Suggested Algorithm

Inasmuch as both optimistic and pessimistic approaches have some weaknesses we must opt the method that is compatible with time aspects and TDB features and has low shortcoming. In optimistic method data item blocking technique is used for inconsistency prevention. This technique decline degree of parallelism and makes deadlock too, so for these reasons the optimistic method is more appropriate as seen in delete and update operation details in the VTR and BTR case, only valid data during valid time interval is changed. Moreover, since query is based on portion of time therefore this interval should be considered for concurrency controller[19,20]. In this case it is better to take into account time interval as granule rather tuples or other data items. In this paper to cope on optimistic weakness we have applied end of transaction marker technique, EOT technique, to release locked resources as soon as possible in the validation phase. To preclude false conflict detection we use time interval for every GT as granule and in this way we can decrease the rate of abortions. Granule state changes are achieved according to portion of time. For instance, in relation CheckedOut after execution of operation "Update(CheckedOut, "CustomerID=C102", 2, 4, TapeNum: T1245)" only valid data during [2, 4] interval are updated thus interval [2, 4] is considered for concurrency controller. So, in BTR case, granule is defined as Granule: (R, gk,T_i) whereas R is relation name, gk is generic key and T_i includes time interval. Optimistic concurrency controller should define four different sets for each operation in Bitemporal Relation Model, one for read set and three for write sets. The sets are RS_i($Read\ set\ of\ T_i$), IS_i($insert\ set\ of\ T_i$), US_i($update\ set\ of\ T_i$) *and* DS_i(Delete set of T_i)[9a,10a,21a]. According to this method, every transaction passes three phases. The first is the time between the read phase which contains the time of requesting data item from DB and the time of coping on local space that is not accessible with other transactions. The second phase is validation time when conflict auditing starts as the method is optimistic. It starts after C_i for each T_i. The third is write phase when real manipulation is done on DB. Afore-mentioned phases make critical section. As each transaction which is executing its last command C_i , concurrency controller must check whether T_i has conflict with other T_j or not. If so, transaction with low priority timestamp should be aborted. For the sake of performance amelioration, validation phase will be commenced right after exiting critical section and end of transaction marker technique, EOT, should be applied to set free locked granule as soon as possible to enhance degree of parallelism. When transaction T_i passes its certification test, concurrency controller marks the last command in RS_j where T_j are transactions not validated yet. To find out conflict detection between $T_i$and T_j , not validated yet, we use $RS_j(T_i)$ whereas RS_j is limited to objects read by T_i from beginning to the end of transaction mark. After receiving messages correspondent to their operations, data item will be eked out to their correspondent set by concurrency controller. Likewise, after receiving suitable message correlated to Rollback(T_i) operation data item will be deleted from RS_i, IS_i, US_i and DS_i. Finally, after receiving suitable message correlated to Commit(T_i) concurrency controller must check whether T_i has conflict with T_i , not yet validated, or not.

3.1 Proof of Conflict Recognition

To prove the conflict detection, formal technique is needed and it is serializable theory [1]. Assume for each transaction T_i, algorithm constructs set of operation O_i . There are four operations on granule x like Read:$R_i(x)$, Insert: $I_i(x)$, Delete:$D_i(x)$,Update:$U_i(x)$ and end of transaction operations either Commit:C_i or Rollback:R_i. Moreover, transaction operations belong to partial order $(T_i, <_i)$ so that :

1. $T_i \subseteq O_i$;that is, all the execution operations should be considered.
2. $R_i \in T_i$iff $C_i \notin T_i$; that is,T_i contains either R_i or C_i.
3. If $t \in \{C_i, R_i\}$ then for each $O_i \in T_i$ then $O_i <_i t$.
4. If there is conflict between O_j and O_k then either $O_j(x) <_i O_k(x)$or $O_k(x) <_i O_j(x)$.

If $T = \{T_1, T_2, \dots, T_n\}$ is the set of all transactions and H is a complete history on T then H is partial order $<_H$ that is :

1. $H = \cup_{i=1}^{i=n} T_i$
2. $<_H \supseteq \cup_{i=1}^{i=n} <_H$
3. For each two conflicting operations p , q $\in H$ either $p <_H q$ or $q <_H p$.

Suppose that T_i and T_j are separate transactions which access to common resource concurrently and T_i is high priority in contrast to T_j(TS(T_i)< TS(T_j)). Only conditions in $D_i \backslash R_j$, $D_i \backslash U_j$, $U_i \backslash R_j$ and $I_i \backslash I_j$ can constitute conflict situation [R].

Claim: Assume p_i andq_j that can be update and delete with $TS(p_i(x)) < TS(q_j(x))$ and have conflict on common data item x. To guarantee for keeping consistency when concurrency controller receives message correlated to operation p_i it should be checked intersection between specified operation set and operation set related with all operations pertained to T_j, not to be validated, that have conflict with T_i whether intersection is empty or not. If any, the transaction with low priority must be aborted. For instance, if $p_i(x)$is delete on data item x and $q_j(x)$is update on x, whenever concurrency controller is checking valid phase for T_i when T_j is not validated yet and has conflict withT_i too then $DS_i \cap US_j = \phi$should be checked. If it is true then soundness is guaranteed otherwise transaction T_j , with low priority, has to be aborted.

Proof: Let T_i be a transaction to be validated and T_j is not validated yet whereas $DS_i \cap US_j \neq \phi$ andH_s is a complete history of transactions and is serial too that is all operations ofT_i, that have conflict with operations of T_j, are before all operations of T_j. A commited history of$H$, which all active and aborted transactions to be neglected nominated *C(H)*, is serializable if it is equivalent to $H_s(C(H))$. Note that T_i is the first transaction to be validated and T_j is not still aborted. For each operations $p_i \in T_i$ and $q_j \in T_j$ in conflict if $p_i <_H q_j$then $p_i <_{C(H)} q_j$. As operations in D_i and U_j are in conflict and T_i should be validated sooner then we must have $D_i <_{C(H)} U_j$. The last point implies U_j is executed sooner than D_i because it was assumed that D_i is not validated until right now therefore $U_j <_{C(H)} D_i$ whereas it is apparent contradiction that is , serializable graph is not acyclic and the history is not serializable. Consequently, the intersection of DS_i and US_j has to be empty[21,22].

Our proposed algorithm is Optimistic_ConcurrencyController,Optimistic_CC in short, receives messages accordance to operation and react appropriately. It is presented with details in pascal pseudo code. Moreover, this algorithm guaranteesto be serializable, consistent and validated. Mocros*RDM, ISM, DLM, UDM, RBM* and *CMM*

are messages for Read, Insert, Delete, Update, Rollback and Commitoperations respectively. The functions that have term "EkeOut" in the first of them during calling the function eke out data item to the relevant set of operation. For example, when function EkeOutRead(T_i, Granule); is called then during the execution data item granule will be eked out to RS_iset. Moreover, Rollback(T_i) deletes all data items from all sets and Commit(T_i) does validation phase of T_i[21]. Procedure Optimistc_Concurrency Controller and sub procedures are shown below:

```
ProcedureOptimistic_Concurrency Controller;
Begin
    While there is more messageDo
ReceiveMessage;
        Case Message of
RDM: ReadMessage (Ti, Granule);
EkeOutRead(Ti, Granule);
            ISM:InsertMessage(Ti, Granule);
EkeOutInsert(Ti, Granule);
DLM:DeleteMessage(Ti, Granule);
EkeOutDelete(Ti, Granule);
       UDM:UpdateMessage(Ti, Granule);
EkeOutUpdate(Ti, Granule);
RBM:RollbackMessage(Ti);
Rollback(Ti);
       CMM:CommitMessage (Ti);
Commit(Ti);
End{Case}
End{While}
End;{ Optimistic_CC }
Procedure Commit(Ti);
Begin
Validation(Ti);
Convert the state of Ti from Active to Commited;
    Add Ti to commited list;
    Drop Ti from Active list;
End;{ Commit }
Procedure Validation(Ti)
Begin
Send “Ok Message" to transaction manager;
(* It is used to distribute the writings on DB *)
     Debut for critical section of Ti .
j := timestamp of the next transaction immediately arriving after Ti.
While j <> "!" Do
    Add the macro "EOTi" to RSj , USj and ISj.
j := timestamp of the next transaction immediately arriving after Tj.
End{While }
Send an appropriate message to transaction manager
(* This message indicates the end of critical section for Ti provided all Ti writings
on DB are achieved. *)
```

```
j := timestamp of the next transaction immediately arriving after T_i.
While j<> "!" Do
If( Conflict(T_i,T_j) is True ) Then
      Delete all items from o RS_j , US_j, DS_j and IS_j.
         Send abort command a_j for T_j to transaction manager and re-execute T_j.
End{If }
j := timestamp of the next transaction immediately arriving after T_j.
End{While }
End{ Validation }
Function Conflict(T_i, T_j): Boolean;
Begin
OutCome:= False;
  (* XS_j(T_i) contains the objects manipulated by T_j until EOT_i *)
  If (DS_i ∩ RS_j(T_i) ≠ ϕ ) OR
     ( DS_i ∩ US_j(T_i) ≠ ϕ ) OR
     ( US_i ∩ RS_j(T_i) ≠ ϕ ) OR
     ( IS_i ∩ IS_j(T_i) ≠ ϕ ) OR
     ( IS_i ∩ RS_j(T_i) ≠ ϕ ) then OutCome := True;
EndIf;
Conflict :=OutCome;
End;
```

4. Performance Assessment OfOptimistic_CC Versus 2PL-pessimistic

To evaluate our suggested algorithm versus pervasive and standard pessimistic 2PL that is being used in existing DBMS applications, we should define our experimental data structure[23]. Our proposed data structure and value range is brought on figure4.

Arrival Rate	Data item frequency	DB location	Transaction size	Read size	Write size	R/W ratio
20 tr. Per sec. To 50 tr. Per sec.	1000 items	RAM to Hard Disk	5 to 10 operations	2 to 100	1 to 50	0.2 to 0.5

Figure4. Experimental data structure

Arrival time is the rate of transaction execution from transaction manager and data item frequency is the number of data item in DB and R/W ratio is a proportion of read operations in contrast to write operations. For example, if T_i contains five operations and one of them is Read and the rest are Write then R/W ratio is 0.2. To simulate exact execution, we consider three events working on hard disk DB and the rate of events will be increased gradually to reach stable results. Figure5 illustrates three events.

Events	Arrival rate	Mean Read Size	Mean Write Size
Event1	20 tr./sec.	20	10
Event2	40 tr./sec.	25	15
Event3	50 tr./sec.	30	20

Figure5. Events for DB on Disk

Figure6 depicts outcome of two executions Optimistic_CC as our suggestion and 2PL-Pessimistic working on data item of Figure1 in BTR case when it reaches to stable point, i.e., event3. The result shows that with increasing the rate of transactions and R/W ratio the execution time in 2PL-pessimistic is approximately growing exponentially, but execution in Optimistic_CC is almost rising linearly.

Metric/Algorithm	**2PL-Pessimistic**					**Optimistic_CC**				
Number of transactions that makes multiprogramming	2	4	6	8	10	2	4	6	8	10
Execution Time	5	10	50	120	500	5	10	15	25	30

Figure6. Execution time regarding to event3(stable point)

5. Conclusion

An Optimistic_Concurrency Controller procedure has been applied accordance to time aspects and temporal database properties containing time referenced data. Moreover, it has been proved that the new algorithm guarantees to be serializable, consistent and validated and detects conflicts in validation phase as soon as possible and releases locked resources by using EOT marker techniques and enhances parallelism by considering time interval as granule rather tuples. Although 2PL-pessimistic uses locking resources technique and our optimistic version uses abortion technique, but result implies that according to low rate of abortion in optimistic method, our Optimistic _CC has high performance. The next generation algorithm must have good trade off provided these algorithms opt suitable granule size and detect conflicts as soon as possible.

6. References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, (1987) "Concurrency control and recovery in BDS", *ADDISON-WESLEY Edition*.

[2] A.Makani and R.Bouaziz, (2010) "Concurrency Control For Temporal Database",The International Journal of Database Management Systems(IJDMS),2(1)

[3] C. S. Jensen, & al., (1998) "The Consensus Glossary of Temporal Database Concepts"—February 1998 Version, O.Etzion, S.Jajodia, S.Sripada, (Eds.), Temporal Databases—Research and Practice,*Lecture Notes in Computer Science*, Vol. 1399, Springer-Verlag, Berlin, pp. 367–405.

[4] C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, and R. Zicari. *Advanced Database Systems*.Morgan Kaufmann Publishers 1997.

[5] S. Abiteboul, R. Hull, and V. Vianu.*Foundations of Databases*. Addison-Wesley 1995.

[6] J. Celko. *Joe Celko's Data and Databases: Concepts in Practice*. MorganKaufman Publishers 1999.

[7] J. Clifford and A. Tuzhilin (eds.). *Recent Advances in Temporal Databases:Proceedings of the International Workshop on Temporal Databases*. Workshopsin Computing Series.Springer-Verlag 1995.
[8] D. Menasce, and T. Nakanishi, (1982) "Optimistic vs Pessimistic control mechanism in databasemanagement systems", *International Journal of Information Systems*, Vol. 7(1).
[9] A. Makni, R. Bouaziz, and F. Gargouri, (2006) "A New Optimistic Concurrency Control Algorithmfor Valid-Time Relations", *Proc. of the 10th IASTED International Conference on SoftwareEngineering and Applications*, November 13-15, Dallas, TX, USA, pp. 117-126. Published byACTA Press, Paper n° 514-143, 2006.
[10] A. Makni, R. Bouaziz, et F. Gargouri, (2007) "Formal Verification of a new Optimistic ConcurrencyControl Algorithm for Temporal Databases", *Proc. of the 16th ISCA International Conference onSoftware Engineering and Data Engineering*, Las Vegas, Nevada, USA, July 9-11, pp. 235-242.ISBN 978-1-880843-63-5.
[11] [11] C. S. Jensen and C. E. Dyreson (eds.). A Consensus Glossary of Temporal Database Concepts—February 1998 Version. [21, pp. 367–405].
[12] C. Bettini, X. S.Wang, and S. Jajodia.A General Framework for Time Granularityand Its Application to Temporal Reasoning.*Annals of Mathematics andArtificial Intelligence*, 22(1–2):29–58, 1998.
[13] [13] C. E. Dyreson and R. T. Snodgrass.Supporting Valid-Time Indeterminacy. *ACM Transaction on Database Systems*, 23(1):1–57, 1998.
[14] J. Clifford, C. E. Dyreson, T. Isakowitz, C. .S. Jensen, and R. T. Snodgrass.On the Semantics of 'Now' in Databases.*ACM Transactions on DatabaseSystems*, 22(2):171–214, June 1997.
[15] C. De Castro, (1998) "On concurrency management in temporal relational databases", *Proc. of theSymposium on SEBD*, pp. 189-202.
[16] C. S. Jensen and R. T. Snodgrass.Semantics of Time-Varying Information.*Information Systems*, 21(4):311–352, 1996.
[17] C. S. Jensen, and D. B. Lomet, (2001) "Transaction timestamping in (temporal) databases", *Proc.Ofthe Conference on Very Large Databases*, Roma, Italy, pp. 441-450.
[18] R. T. Snodgrass. The Temporal Query Language Tquel.*ACM Transactions on Database Systems*, 12(2):247–298, June 1987.
[19] M. Finger, and P. McBrien, (1996) "On the semantic of 'current time' in temporal databases", *Proc.of the Symposium on Databases*, Sao Carlos, Brasil, pp. 324-337.
[20] M. Finger, and P. McBrien, (1997) "Concurrency Control for Perceivedly InstantaneousTransactions in Valid-Time Databases", *Proc. Of the Conference on Temporal Representation andReasoning*, pp. 112-118,.
[21] E. Brinksma, A. Mader, and A. Fehnker, (2002) "Verification and Optimization of a PLC ControlSchedule", *International Journal on Software Tools for Technology Transfer*, Vol. 4(1), pp. 21-33.
[22] A. Makni, R. Bouaziz, and F. Gargouri, (2006) "A New Optimistic Concurrency Control Algorithmfor Valid-Time Relations", *Proc. of the 10th IASTED International Conference on SoftwareEngineering and Applications*, November 13-15, Dallas, TX, USA, pp. 117-126. Published byACTA Press, Paper n° 514-143, 2006.
[23] A. Makni, R. Bouaziz, et F. Gargouri, (2007) "Performance Evaluation of an OptimisticConcurrency Control Algorithm Ensuring Strong Consistency for Transaction Time Relations",*Proc. of the International Conference on Enterprise Information Systems and Web Technologies*,Orlando, FL, USA, July 9 - 12, pp. 258-265.