An Innovative Method for Organizing Incompatible Information at Ideal Time

Ali Mohammed Saleh Ahmed

University of Diyala- Internet and Computer Center

dr.alisaleh80@gmail.com

Received: 12 March 2017 Accepted: 14 May 2017

Abstract

This study aims at analyzing the problems of transaction management in heterogeneous real-time information systems. It is proposed the use of an algorithm to resolve conflicts in the vague the deadlock transactional situation using construction and reduction of special directed bipartite wait-for graph. Therefore, a reduction algorithm was applied and presented in this article to the count of expectation of transactions G provides the required speed in the conditions of impasses at management of transactions in the heterogeneous distributed information systems of real time. This algorithm proved its efficiency in transaction management.

Key words: database, transaction, wait-for graph, deadlock condition

طريقة مبتكرة لتنظيم المعلومات غير المتجانسة في الوقت الحقيقي

علي محمد صالح احمد

جامعة ديالى - مركز الحاسبة والانترنت

ألخلاصة

تهدف هذه الدراسة إلى تحليل مشاكل إدارة المعاملات في نظم المعلومات الغير المتجانسة في الوقت الحقيقي، ويقترح استخدام خوارزمية لحل النزاعات في ظروف وضع الجمود للمعاملات باستخدام البناء والحد من الموجه الثنائي-wait for graph لذلك، تم تطبيق خوارزمية الحد وقدم في هذه المقالة عدد من التوقعات للمعاملات، ويوفر G السرعة المطلوبة في ظروف الطرق المسدودة في إدارة المعاملات في نظم المعلومات الموزعة الغير المتجانسة في الوقت الحقيقي، وأثبتت هذه الخوارزمية كفاءتها في إدارة المعاملات.

الكلمات المفتاحية: قاعدة بيانات، المعاملات، wait-for graph ، حالة الجمود

Introduction

The creation of new means and methods of software for transaction management systems in heterogeneous information systems, providing increase in the quality level and efficiency of access to the distributed data, acquires the increasing relevance in the conditions of creation and combining DBMS of real time [1]. While transactions with blocking work there can be a deadlock situation in operational conditions of parallel data exchange, that is a situation at which both transactions wait for each other and cannot come to end. Such condition results in lack of a standard exit from a deadlock, therefore it needs to be distinguished and eliminated. Methods of resolving a deadlock is to roll back a transaction (victim-transaction) so that other transactions continue their work. After resolving the deadlock, the transaction selected as a victim shall be restarted.

The main problems which arise in case of parallel execution of transactions can be united into four types[3]: unsaved changes (the situation occurs if two transactions change simultaneously the same record in DBMS); problems of the intermediate data (the situation arises in case of formation of intermediate data on an object, at a stage of executed parallel

transaction until its rollback); problems of uncoordinated data (the situation arises because of data tuple change by a transaction and this data was already read by a parallel transaction); problems of ghost-lines or phantom-lines (the situation arises in case of "invasion" of parallel transaction within change of the actual result).

Methods and Procedures of Transaction Coordinator

It is required to develop a certain procedure of coordinated performance of parallel transactions. This procedure has to satisfy the rules given in the text below:

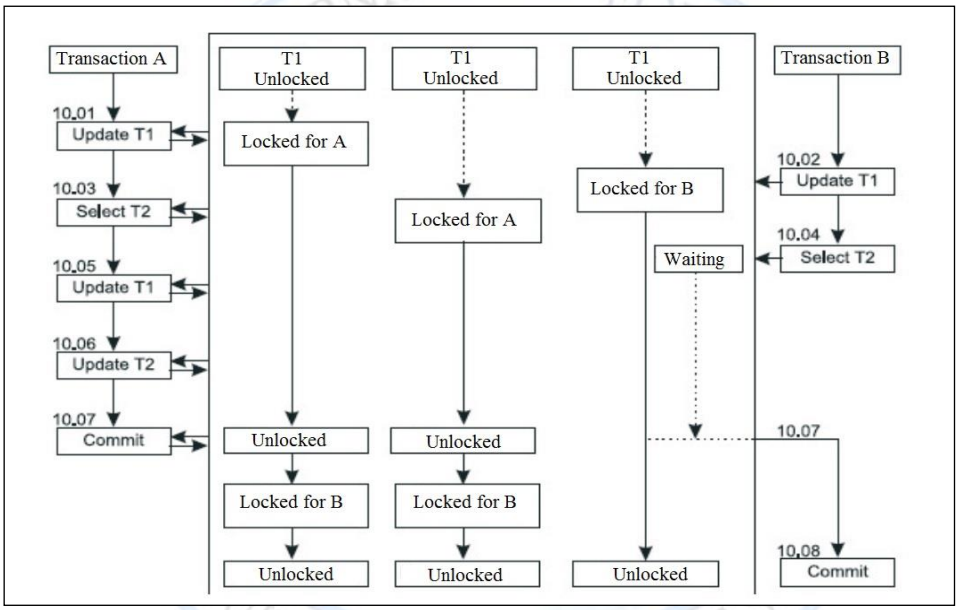

Figure 1: Blocking at simultaneous performance of two transactions

During transaction execution, the user sees only coordinated data (the user shouldn't see uncoordinated intermediate data). [1] When in homogeneous DBMS two transactions are executed parallel, the database supports with guarantee the principle of independent execution of transactions, which states that results of transaction execution will be same as if transaction 1 would be executed first, and then transaction 2 or vice versa, transaction 2 the first, and then transaction 1 would be executed.

Such procedure is called serializing the transactions. Actually, it guarantees that each user (a program, a request) addressing the database works with it as if there are no other users (programs, requests), addressing the same data along with it [2].

Let's consider serializing the transactions in a real-time DBMS conditions. Sequential execution of transactions would be the simplest, but such plan is not optimal conditions because of existence of more flexible methods of management of parallel access to databases. The most widespread mechanism, which is used by commercial DBMS for practical implementation of serializing the transactions, is the blocking mechanism. The simplest option is blocking of an object for all the time of transaction execution. A similar example is reviewed in figure 1. Two transactions conventionally called A and B operate with three tables: T1, T2 and T3. When transaction starts to work with an object it blocks this object, which becomes inaccessible to all other transactions until the end of the transaction, which has blocked ("captures") this object. After transaction completion, all objects blocked by it get unblocked and become available to other transactions. If transaction addresses the blocked object, it remains pending until this object gets unblocked then it can continue processing this object. Therefore, transaction B waits for unlocking of table T2 by transaction A. Above rectangles there is conditional time of operations execution.

Generally, when performing a transaction gets exclusive access to DB objects with which it works. In this case, other transactions don't get access to DBMS objects until the end of transaction. Such mechanism really eliminates all problems listed above: unsaved changes, unconfirmed data, uncoordinated data, and phantom lines. However, such blocking creates new problems - a delay of transaction performance because of blocking [3]

Considering the existing types of the conflicts between two parallel transactions. The following types can be separated:

- W-W - transaction 2 tries to change the object changed by not ended transaction 1;
- R-W - transaction 2 tries to change the object read by not ended transaction 1;
- W-R - transaction 2 tries to read the object changed by not ended transaction 1.

The blockings called also synchronizing object can be applied to different type of objects. The greatest object to lock can be entire database; however, this type of lock will make a DB

unavailable to all applications, which work with this DBMS. The next type of subject for blocking are tables. Transaction, which works with the table, blocks it for all the time of transaction execution. This type of blocking is more preferable than previous because it allows carrying out in parallel transactions, which work with other tables.

In a number of the DBMS, lock at page level is realized. In this case, the DBMS locks only individual pages on a disk when transaction addresses them.

Table 1: Rules of application of hard and soft blocking of transactions

		Transaction B		
		Unlock	Soft unlock	Hard unlock
Transaction A	Unlock	Yes	Yes	Yes
	Soft unlock	Yes	Yes	No
	Hard unlock	Yes	No	No

This type of blocking is softer and allows different transactions to work even with the same table if they address different pages of data. In some DBMS, blocking at the level of lines is possible; however, such mechanism of blocking requires additional costs on support of this type of blocking. For increase in parallelism of transactions execution, the combination of different types of synchronizing captures is used [1]. Two types of blocking are considered (synchronizing captures):

- The joint mode of blocking – the soft or divided blocking known as S (Shared). This mode means the dividable capture of object and is required for performance of object reading operation. The objects thus blocked don't change during transaction execution and are available to other transactions also, but only in reading mode;
- The exclusive mode of blocking - hard, or exclusive, blocking knows as X (exclusive). This mode of blocking assumes exclusive capture of object and is required for performing of entering, removal and modification operations. The objects blocked by this type of blocking actually remain in the exclusive mode of processing and are inaccessible for other transactions until completion of work of this transaction.

Objects capturing by several transactions are compatible for reading, namely several transactions are allowed to read the same object, object capture by one transaction for reading is incompatible with capture by other transaction of the same object for record, and captures

of one object by different transactions for record are incompatible [2]. Rules of compatibility of captures of one object by different transactions are presented in the table.

In the example, it is considered that transaction A locks first the object, and then transaction B tries to access it.

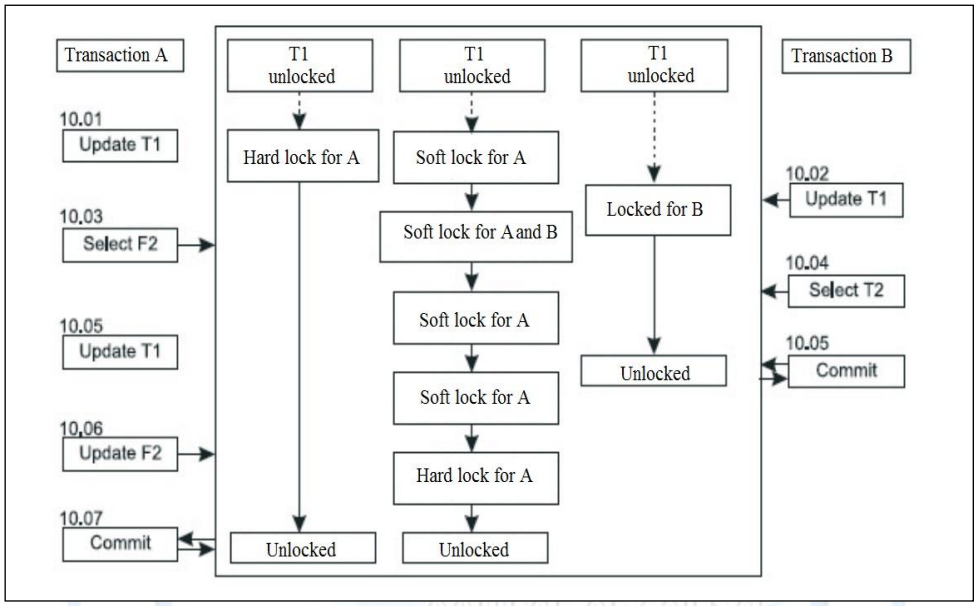

Figure 2: Use of hard and soft blocking

In figure 2 is given the earlier example with performance of transactions 1 and 2, but different types of blocking are considered. The figure shows that application of soft blocking to table 2 by transaction 1 provides substantial reduction of time of transaction 2 performance. Now transaction 2 doesn't wait for the end of transaction 1 and therefore finishes the operation much earlier that gives sufficient efficiency within the homogeneous information environment of DBMS. Unfortunately, application of different types of blocking leads to a problem of deadlocks, which has arisen by consideration of performance of parallel processes in operational environments and also has been connected with management of the divided (shared) resources.

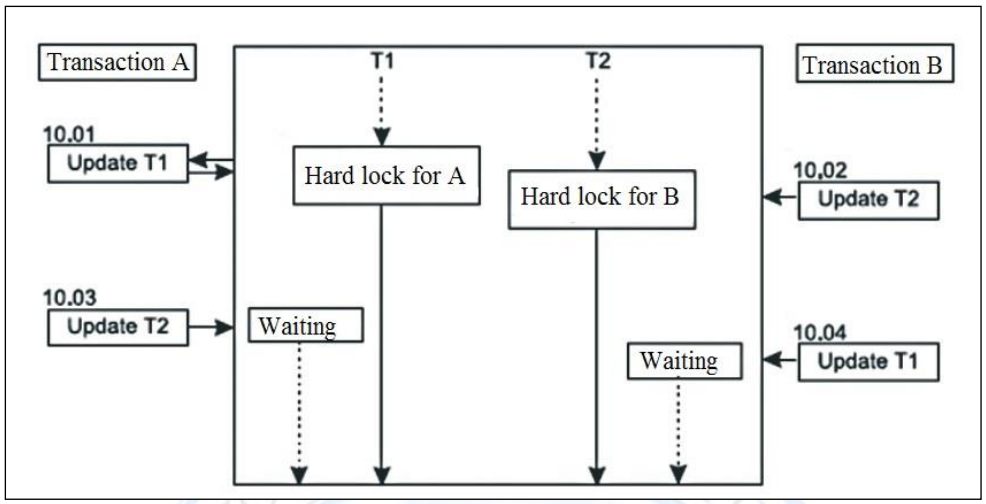

Figure 3: Mutual blocking of transactions

Here is an example. Let transaction A first to block hard the table 1, and then to block hard the table 2. Transaction B, on the contrary, at first blocks hard the table 2, and then blocks hard the table 1. If both of these transactions have begun working at the same time, then after performance of modification operations by the first subjects of each transaction both of them will pend infinitely: transaction A will wait for completion of work of transaction B and unblocking of table 2, and transaction B will wait also without results for completion of work of transaction A and unblocking of table 1 (figure 3). Situations can be much more difficult [2]. The number of mutually blocked transactions can be much more. The basis for deadlock detection is the implementation of operations in the transaction wait-for graph. A transaction wait-for graph is the directed bipartite graph G= (W, E) in which all tops are grouped into two types - the tops corresponding to U transactions and the tops corresponding to V blocking subjects. At the same time performance of the following conditions is important U^V = W, |U |> 0, |V |> 0 so no peak in U isn't connected to peaks in U and any top in V isn't connected to peaks in V. In the considered graph the directed lines S connect only peak-transactions to peak-objects. The directed line from peak-transaction to Si peak-object exists if and only if for this transaction there is a satisfied blocking of this object [1]. The directed line from Si peak-object to top of transaction Sj exists if and only if when this transaction expects satisfaction of blocking inquiry for this object. It is easy to show that in system there is a deadlock condition in that and only that case when in transaction expectation graph G there is

at least one cycle. The simplest example of the transactions expectation graph with a cycle is shown in figure 4.

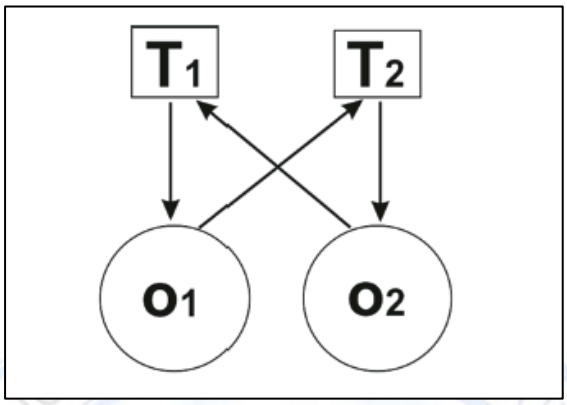

Figure 4: A situation of the synchronizing deadlock between transactions T1 and T2

Let's consider a situation of the synchronous deadlock: transactions T1 and T2 set exclusive locks of objects O1 and O2 respectively; after this T1 requires joint lock of object O2, and T2 requires joint lock of object O1; no one of these requirements of lock can't be satisfied, therefore, no transaction can proceed; therefore, exclusive locks of objects will never be removed, and requirements of joint locks won't be met.

As deadlocks are possible and no natural outcome from lockup exists, these situations need to be found and synthetically eliminated within the constructed coordinator of transactions of the general information system [3]. To detect the deadlock conditions a transaction wait-for graph is periodically formed in transactions of which cycles appear. By development of the mathematical description of the transactions coordinator in case of the synchronous deadlock, a non-standard approach to a reduction of the directed bipartite wait-for graph is used. Let's explain the order of application of reduction algorithm to the transaction wait-for graph G in conditions of exclusive blockings. The initial condition of graph is presented in figure 5.

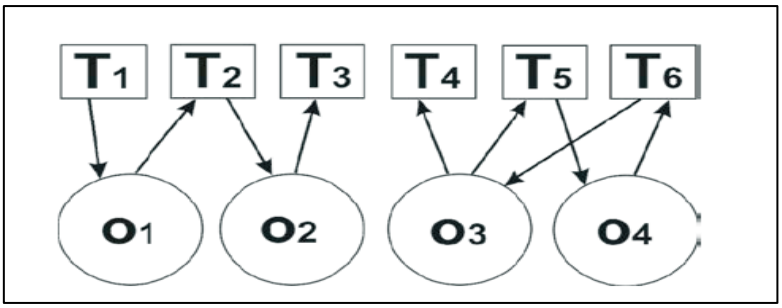

Figure 5: Initial condition of expectation count

In this case, the procedure of reduction is that first of all from the wait-for graph G all those directed lines are being deleted which go from top-transactions and to which no directed lines from top-objects enter. Besides, those directed lines are removed which enter to peak-transactions and from which no directed lines conducting to peak-objects start from [3]. For those peak-objects for which there is no entering directed lines but there are outgoing ones, orientation of one of outgoing directed lines (chosen arbitrarily) changes to an opposite [2]. After performance of the first step of a reduction of mirror algorithm the subsequent steps are being performed until the removal of directed lines stops.

In case of need of deadlock destruction, the build transaction coordinator chooses a transaction victim Tk in a random way and makes its roll-back, and afterwards resends it. In fig. 6, the reduction implementation algorithm in wait-for graph in conditions of transaction management by a real-time DBMS is provided [3].

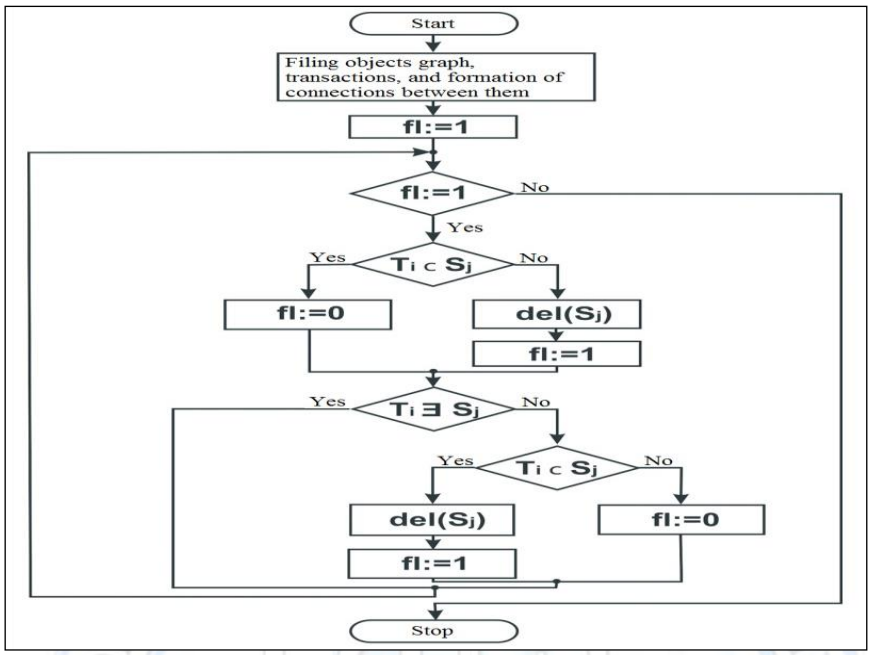

Figure 6: The block diagram of an algorithm of a reduction in the column of expectation

Results

The following results were obtained:

1. Architecture of web-based applications for multiprocessor and cluster solutions that are scalable and provides high performance when inter-module interaction in the database.
2. Hierarchical model of access to data and internal information exchange in a web-based interface that provides rapid generation of data based on queries to the database information environment.
3. The concept of transaction was extended to information systems in general, not just to DBMS as a part of such systems.
4. This solution provides enhanced operational efficiency allowing several applications to work jointly.
5. This algorithm proved its efficiency in transaction management.

Conclusion

Application of the reduction algorithm presented in this article to the count of expectation of transactions G provides the required speed in the conditions of impasses at management of transactions in the heterogeneous distributed information systems of real time. All this absolutely meets requirements of creating the considered information system. Success of creating an effective system of transaction management processes in the electronic information system will provide a sufficient level of performance form user requests Web-based applications of the system from a single database service.

References

1. Tsimbal, A.A. Technologies of creation of the distributed systems. For professionals [Text] / A.A. Tsimbal, M. Anshina. St.Petersburg: Publishing House «Petersburg», 2003. – p.576
2. Rikov, S.A. Management of the heterogeneous distributed objects of information systems of real time [Text]: monograph / S.A. Rikov, V.L. Burkovsky, A.A. Golikov. – Voronezh: Federal State-Funded Educational Institution Higher Education «Voronezh State Technical University», 2012 – p.190
3. Burkovsky, V.L. Simulation and algorithmization of control of heterogeneous databases in the distributed information systems [Text]: monograph / V.L. Burkovsky, A.N. Dorofeev, S.V. Seminin – Voronezh: VSTU, 2013 – p.71
4. Ozsu M.T., Valduriez P. Distributed Database Systems: Where Are We Now, Computer (August 1991), Vol.24 (No.8): 68-78.
5. Sheth A.P., Larson J.A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3): 183-236, September 1990.
6. Smith J.M., Bernstein P.A., Dayal U., Goodman N., Landenrs T., Lin K.W.T., Wong E. Multibase – integration heterogeneous distributed database systems. In National Computer Conference, volume 50 of AFIPS Conference Proceedings, pages 487-499, 1981.
7. Tesch T., Wasch J. Global Nested Transaction Management for ODMG – Compliant Multi-Database Systems. In Proceedings of the Sixth International Conference on

Information and Knowledge Management (CIKM'97), Las Vegas, Nevada, November 10-14, 1997.

8. Optimal Scheduling of Queueing Networks with Switching Times Using Genetic Algorithms / S. Podvalny, V. Burkovsky, S. Semynin, S. Titov. WSEAS Transactions on systems. Issue 5, Vol. 5, Prague, Czech Republic, 2006 P. 1060-1065.
9. Rusinkiewicz M., Georgakopolous D. Multidatabase transactions, impediments and opportunities. In COMPCON Spring 91Digest of Papers, pages 137-144, 1991.
10. Thomas G., Thompson G.R., Chung C., Barkmeyer E., Carter F. Heterogeneous distributed database systems for production use. ACM Computing Surveys, 22(3): 237-266, September 1990.
11. Mchrotra S., Rastogi R., Korth H.F., Silberschatz A. The Concurrency Control Problem in Multidatabases. Characteristics and Solutions, Proc. Of ACM-SIGMOD International Conference on Management of Data, 1992, Pg.: 288-297.
12. Hussein K. AL-Khafaji , Noora A. Al-Saidi. A New Algorithm to Preserve Sensitive Frequents Itemsets (APSFI) in Horizontal or Vertical Database. Engineering & Technology Journal. 2013, p 755-769.
13. Sama Salam Samaan, Design and Implementation of a Pharmaceutical Inventory Database Management System. Al-Khwarizmi Engineering Journal, 2017 p 118-128.
14. Ghassan H. Abdul-Majeed, Alaa Kadhim F., Rasha Subhi Ali. Retrieving Encrypted Query from Encrypted Database Depending on Symmetric Encrypted Cipher System Method. Diyala Journal For Pure Science. 2016 p. 183-207.