Concurrency Control for Global Transaction Management in MDBSs

Kyu-Woong Lee[1] and Seog Park[2] and Gil-Rok Oh[1]

[1] Computer & Software Tech. Lab., ETRI,
Yusong P.O. Box 106, Taejon, 305-600, KOREA
{leekw, groh}@etri.re.kr
[2] Dept. of Computer Science, Sogang University,
C.P.O. Box 1142, Seoul, 100-611, KOREA
spark@dblab.sogang.ac.kr

Abstract. The objectives of global transaction management in multidatabase systems(MDBS) are to avoid the inconsistent retrievals and guarantee the global serializability under the existence of *indirect conflict* which is unknown to to the global transaction manager(GTM). Many researches have shown that it is difficult to design the global concurrency control method because of local autonomy. In these method global transactions have a few opportunities to be executed concurrently. We concentrate our attention on 1) investigation into the more accurate indirect conflict situation and 2) supporting the higher concurrency degree by using the concept of global integrity constraints. We define the multidatabase transaction model and then propose the concurrency control protocols. In our method the more global transaction can be concurrently executed, since the refined boundary of possibility of indirect conflict is offered.

1 Introduction

The GTM has the responsibility for maintaining the global consistency of MDBS. The GTM however cannot take any kind of helpful information from local database systems to adjust the global serialization order. Many techniques in managing transactions for MDBS environments have been researched. These vary in degree to which they violate local autonomy and also in the degree of concurrency that they provide to users. The major difficulties addressed in these researches are to serialize the global transaction under the existence of indirect conflicts which is unknown to the GTM. Traditional concurrency control methods are not directly applicable in MDBS because of these indirect conflict. We briefly examine the definition of indirect conflict.

Definition 1. The global transaction G_i and G_j are in indirect conflict in global schedule S if and only if there is a local transaction sequence $L_1, L_2, \ldots, L_r$, such that G_i is in direct conflict with L_1 , L_1 is in direct conflict with L_2, ..., finally, L_r is in direct conflict with G_j. □

In order to resolve these indirect conflicts, previous researches provide global concurrency control methods with the lower degree of concurrency by using forced data conflicts between global transactions. They did not figure actual indirect conflict situation. In this paper we adopt the characteristics of *global integrity constraints* to achieve the higher degree of concurrency. A consistent global database state is fragile if a local transaction can be executed by the LDBS that is unaware of inter-site integrity constraints. Thus we investigate the more accurate indirect conflict cases and present the multidatabase transaction model by using the property of global integrity constraints.

The rest of paper is organized as follows. In next section, we describe the related works on transaction management of MDBS and their problems. In Section 3, we propose *site-locking* method that guarantees the global serializability of MDBS. We define the new kinds of lock type for acquiring a site and explain its protocol. Section 4 concludes the paper.

2 Research Backgrounds

2.1 Previous Works

Several researches have been proposed in order to ensure the serializability in MDBS environment [BST90, DELO89, GRS93, MRKS91]. These methods are classified into three groups depending on their strategies and assumptions as follows:

- Violation of local autonomy
  These methods violates local autonomy in some degree in order to control local transaction [AGMS87, EH88]. These methods have impractical assumptions. It has proved that the global serializability is not ensured in these methods [DELO89].
- Relaxation of correctness criteria
  The solutions in this category introduce the notion of *quasi serializability* by using the hierarchical nature of global concurrency control scheme [DEK91] and *two level serializability* [MRKS91, MRKS92]. These methods assume that there is no value dependency between subtransactions within a global transaction or there is weaker form of value dependency. Under these assumption, they guarantee the relaxed criteria rather than the conflict serializability(CSR). *Two level serializability* especially can be ensured in the specific transaction model, and moreover, there is no value dependency between local and global data items.
- Forcing some restriction on LDBS
  The techniques in this category force the each participating LDBS to be restricted on execution of transaction. [WV90] proposes the *2PC agent method* in which each LDBS uses *strict two phase locking*. The fact that local schedule has the strictness is not sufficient to preserve the global serializability. [BGRS91] requires that each LDBS produce the *rigorous schedule*. Rigorousness has all the properties of *strictness* and the extra property that no data

Table 1. Summary of Previous Works

related works	violation of local autonomy	correctness criteria	ensuring global serializability	restriction on global transaction	preserving global IC
[AGMS87]	√	CSR		√	
[EH88]	√	CSR		√	
[DEK91]		QSR	△	√	△
[MRKS91] [MRKS92]		2 Level SR	△	√	△
[WV90]		CSR		√	
[BGRS91]		CSR	√	√	
[BST90]		CSR	√	√	
[GRS93]		CSR	√		

√ : positive △ : neutral

item may be written until the transaction, which previously read it, either commits or aborts. The rigorous schedule is the smaller subset of serializable schedule than one of traditional concurrency control scheme. In other words, the global transaction is restricted to access some data items that may cause the indirect conflict [WV90, BST90, BST92]. In practical cases, however, the major purpose of MDBS is to freely access the distributed data. [GRS93] proposed *optimistic ticket method(OTM)* which requires extra data item, called *ticket* per a LDBS. The conflicting order for the ticket reflects relative serialization order of global transactions. The global serializability is ensured by the forced data conflict on ticket, but concurrency degree of OTM is lower than conventional distributed concurrency control method.

Table 1 shows the summary of previous works for the global transaction management in MDBS. Previous researches for global concurrency control have the following difficulties.

- They did not figure the actual indirect conflict cases. They suppose that the global transaction is indirect conflict if any other global transaction is concurrently executed in the same site.
- They provide lower concurrency degree that is similar to the performance of a serial execution.
- They have non-realistic assumptions such that there is no inter-site constraints or value dependency between local and global data items.

2.2 Research Motivation

Each LDBS in MDBS is designed and implemented independently and also defines certain local integrity constraints(LIC) among data items within a single site. However, as a number of various DBMSs are integrated into an MDBS, global inter-site constraints are introduced. These distributed integrity constraints arise naturally whenever the data that is semantically related is stored in different local database systems. Each LDBS is unaware of these global integrity constraints. The most fundamental issue of global integrity constraints in MDBS are how and where the global integrity constraints is maintained without the violation of local autonomy.

A logically consistent global database state may become inconsistent if the data is modified without maintaining the global integrity constraint. For example, if a data item is defined in global integrity constraints, the update on it through the interface of $LDBS$ may cause an inconsistent database state. The data item especially may be replicated at different sites. In this case, the consistency of its replicated versions cannot be maintained by LDBS, because LDBS is unaware of replicated version at different site. Since the LDBS does not have the capability to maintain the global integrity constraint, the data item which is defined in the global integrity constraint should be managed by the GTM. Hence, restriction on the local transaction is necessitated in MDBS.

3 Site-Locking Protocol for Global Concurrency Control

3.1 Integrity Constraints and Transaction Model

In MDBS environment, the integrity constraints are classified into local and global integrity constraint. The local integrity constraints are pre-existing integrity constraints which are maintained by LDBS, and the global integrity constraints are newly defined by MDBS according to the integration process.

The introduction of inter-site constraints enable us to partition the set of data item at a site i, D_i, into local data items, LD_i, and global data items, GD_i, such that $LD_i \cap GD_i = \phi$ and $D_i = LD_i \cup GD_i$[MRKS91]. Furthermore, if there is an integrity constraints between the data item $d_i \in D_i$ and $d_j \in D_j$,$i \neq j$, then data items d_i and d_j are global data items in GD_i and GD_j, respectively. Therefore, we partition the data items of MDBS in two groups, as follows [LP97].

- *Global Data Item :*
  the set of data items which is defined in the global integrity constraints
- *Local Data Item :*
  the set of data items which is defined in the local integrity constraints

In our work, we prohibit the local transaction from updating on global data item without the knowledge of global integrity constraints. This restriction enables us to easily maintain global integrity constraints and reduce cost of verifying the global consistency. The local transaction, however, is not restricted to read the data item. The local transaction can read both local and global data items. Some researches propose the method that the global transaction is restricted to read and update the data item and they assume that there are no inter-site constraints [DEK91, MRKS91]. As described in Section 2, some approaches propose the transaction model in which the data set that can be updated by global or local transaction is completely separated [BGRS91, BS92, BST92]. In practical cases, the global transaction cannot be imposed on updating the data item. These restrictions cannot be adopted in general MDBS system, because the global transaction must be free to access the data item. Hence, in our work, global transaction is free to access data items.

We define our multidatabase transaction model called *Global-Free Transaction Model* for maintaining global integrity constraints as described in Table 2.

Table 2. Global-Free Transaction Model for MDBS

Transaction \ Data		Data Item: Local Data	Data Item: Global Data
Local Transaction	Read Operation	◯	◯
	Write Operation	◯	×
Global Transaction	Read Operation	◯	◯
	Write Operation	◯	◯

◯ : possible × : impossible

3.2 Ensuring the Global Serializability

We describe the situation that the global transaction G_i cannot be serialized with respect to other global transaction G_j. If there is a direct conflict between global transactions and at least one indirect conflict between them(Figure 1 (a)), the global serializability cannot be ensured. Similarly, if there exists the indirect conflict at two or more sites(Figure 1 (b)), the global serializability cannot be ensured. An indirect conflict consists of at least two direct conflicts between local and global transaction as shown the Figure 1. Thus, we need to investigate the accurate situation of direct conflict between global and local transaction.

In our *Global-Free* transaction model, we can easily find the situation that there is no direct conflict between the local and global transaction. The Table 3 shows all cases of direct conflicts between the local and global transaction, by classifying the operation of transaction. In Table 3, "◯" means that the direct conflict cannot occur between corresponding two operations. In these cases, two operations are both read operations or they are executed on the distinct data group. On the other hand, "×" denotes that the direct conflict may exist. Such a direct conflict between local and global transaction may cause the indirect conflict between global transactions. The indirect conflict consists of at least

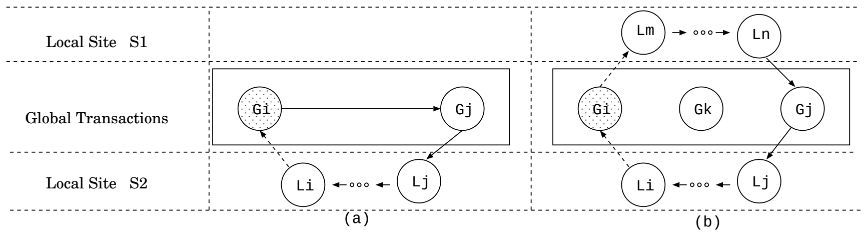

Fig. 1. Globally Non-serializable Schedule

Table 3. Direct Conflict between the Local and Global Transaction

			Global Transaction			
			Read Operation		Write Operation	
			Global data	Local data	Global data	Local data
Local Transaction	Read Operation	Global data	○	○	×	○
		Local data	○	○	○	×
	Write Operation	Local data	○	×	○	×

○ : no direct conflict × : direct conflict

two direct conflicts between local and global transaction. Therefore, if one of the direct conflicts can be resolved or prevented, the global serializability is ensured in multidatabase transaction management.

3.3 Lock Operation for Accessing the Site

In this section, we propose a *site-locking* global concurrency control method that use the lock operation for accessing the site. The basic idea of site-locking method is that a global transaction cannot execute concurrently with other global transactions if it has the possibility of indirect conflict. We enhance the degree of concurrency by investigating the more actual indirect conflict situations.

In Table 3, we list all possible cases of direct conflict between the local and global transaction. However, MDBS cannot know which type of local transaction is executed in the site, when the global subtransaction is submitted to the LDBS in that site. Hence, The GTM must guarantee global serializability by controlling only the global transaction.

Definition 2. The *indirect conflicting operation* is the operation of global transaction that causes the indirect conflict with the other global transaction. Formally speaking, if the operation $p_i(x)$ of global transaction G_i is in indirect conflict with the operation $q_j(y)$ of other global transaction G_j, $i \neq j$, then two operations $p_i(x)$ and $q_j(y)$ are *indirect conflicting operations*. It is not necessary that two data items, x and y, are distinct. If two operations $p_i(x)$ is indirect conflict with $q_j(y)$, $x = y$, then operations p and q must be both read operations. Otherwise, the two operations $p_i(x)$ and $q_j(y)$ are *direct conflicting operations*. □

If we can determine the serialization order of *indirect conflicting operations* of global transactions, the global serializability is guaranteed.

Our global concurrency control method uses the lock operation for indirect and direct conflicting operations, like basic 2PL[BHG87]. In Table 3, we can easily find that the read operation of global transaction that is executed on the global data item is not in direct conflict with any local transactions. Therefore, the read operation of global transaction for global data item cannot be the *indirect conflicting operations*. The remainder operations of Table 3 may cause the indirect conflict with other global transactions because it may be in direct conflict with the local transaction.

To present our *site-locking* method, we need some notations. Locking granularity is a site, rather than a data item. An operation of global transaction is submitted to a site either for *reading* or *writing* a data. Moreover, a read operation is submitted to a site for reading either the *global* or *local data item*. We associate three types of locks with sites : *Read_Global(RGL)*, *Read_Local(RLL)*, and *Write(WL)* locks. We use $RGL_{Gi}(S_x)$ (or $RLL_{Gi}(S_x)$) to indicate that the global transaction G_i has obtained a read lock for global(or local) data item on site S_x. Similarly, we use $WL_{Gi}(S_x)$ to indicate that the global transaction G_i has obtained a write lock on site S_x. We use $RGU_{Gi}(S_x)$(or $RLU_{Gi}(S_x)$) to denote the operation by which G_i releases its read lock for global(or local) data item on the site S_x, and similarly, $WU_{Gi}(S_x)$ to denote the operation by which G_i releases its write lock on the site S_x. A global transaction must acquire one of corresponding lock before its operation is submitted to the site.

Let the P and Q be an arbitrary type of lock operation and G_i and G_j are global transactions which access the same site S_x. P and Q are of conflicting types if one of following conditions are satisfied.

i) Either P or Q is a $Write$ lock(WL), and the other is *Read_Local* lock(RLL), or $Write$ Lock(WL).

ii) Either P or Q is a $Write$ lock(WL), and the other is *Read_Global* lock(RGL), and also G_i and G_j access the same data item.

We define the *site-conflict* between global transactions.

Definition 3. Two locks $P_{Gi}(S_x)$ and $Q_{Gj}(S_x)$ are *site-conflict* if $G_i \neq G_j$ and P and Q are of conflicting types. □

By Definition 3, we present the *site-lock compatibility matrix* for controlling indirect conflicts, as shown in Table 4. Two locks are *site-conflict* if there may be

Table 4. Site-Lock Compatibility Matrix for MDBS

G_i(owner) / G_j(requester)		Read Lock: *RGL* (Global Data)	Read Lock: *RLL* (Local Data)	Write Lock *WL*
Read Lock	*RGL* (Global Data)	○	○	⊗
Read Lock	*RLL* (Local Data)	○	×	×
Write Lock *WL*		⊗	×	×

○ : shared mode × : *Site-Conflict* mode ⊗ : shared or *Site-Conflict* mode

indirect conflict between two global transactions or there is a direct data conflict between them.

3.4 Protocol of Site-Locking

We illustrate the more detailed rules and site-locking protocol. To maintain global serializability, *site-locking protocol* must ensure that the subtransactions of each global transaction have the same relative serialization order in their corresponding LDBS. It is the basic idea of site-locking protocol that the relative serialization order of the subtransactions at each LDBS is reflected in the order of acquiring the site lock operation.

Site-locking protocol processes a global transaction G_i as follows. Before an operation of global subtransaction G_{ix} is submitted into a site S_x, the GTM must acquire the associated site-lock of site S_x. The site-lock which is once acquired by the global transaction G_i is released only after the G_i has been globally committed. Our site-locking protocol is almost the same as basic two phase locking, except that the three types of site-lock is used and the locking granularity is a site rather than a data item. We present our rules according to which the GTM manipulates its site-locks.

Rule 1 : When GTM receives an operation of global subtransaction G_{ix} for the site S_x, the GTM determines the corresponding the site-lock type and tests if the requesting site-lock operation is *site-conflict* with some site-lock operations that are already set at a site S_x, according to Table 4. If so, the requesting operation is delayed until the owner releases the site-lock. If not, then the GTM sets the associated site-lock and sends the operation of global transaction to the LDBS at the site S_x.

Rule 2 : Once the GTM has set a site-lock on the site S_x for the transaction G_i, it may not release that site-lock at least until the global transaction G_i has globally been committed.

Rule 3 : Once the GTM has released a site-lock for a global transaction, it may not acquire any more site-locks for that global transaction.

Rule 1 prevents two global subtransactions from concurrently accessing the same site in *site-conflict* mode. Therefore, the site-conflicting operations of global transactions are scheduled in the same order in which the corresponding site-locks are acquired for the site. As illustrated in Section 2, the basic problem of MDBS is that the GTM cannot determine the serialization order of global transactions at local sites. However, by the *Rule 1*, we can determine the serialization order of indirect conflicting global transactions. *Rule 2* and *Rule 3* are for the growing phase and shrinking phase of basic two phase locking method, respectively.

Our *site-locking* protocol is summarized as follows.

Decomposing the global transaction : The global transaction G_i is decomposed into several global subtransactions.

Requesting site-lock : Before the global subtransaction G_{in} is sent to the local site, it requests the corresponding *site-lock*.

Sending the subtransaction : If G_{in} acquire the *site-lock*, it is sent to the local site. Otherwise, it waits until the corresponding *site-lock* is acquired.

Processing in LDBS : G_{in} that is sent to the local site is controlled and executed by LDBS as a general local transaction.

Releasing the site-lock : If all global subtransactions enter their *prepared-to-commit* state, G_i commit and all of their *site-lock* is released.

According to our proposed *site-locking protocol*, a global transaction that may have an indirect conflict with other global transaction can be serialized and global serializability is guaranteed.

4 Conclusion

Our goal in this paper has been to propose the global concurrency control in MDBS which provides the higher concurrency degree. Primary difficulties in previous researches are 1) the lack of concurrency control protocol that ensure the global consistency in the presence of unknown updates by the local transaction, and 2) the insufficient investigation of indirect conflict cases. These problems introduce numerous unnecessary data conflict between global transactions as well as inconsistent global database state. In this paper, by using the characteristics of global integrity constraints, *site-locking protocol* is proposed to guarantee the global consistency and enable the more global transactions to execute concurrently. We have a plan to prove and analyze our proposed method and have the more detail comparative experiments on simulation model. Our work can be extended to various concurrency protocol in order to adapt to diverse distributed applications.

References

[AGMS87] Rafael Alonso, Hector Garcia-Molina, and Kenneth Salem. "concurrency control and recovery for global procedures in federated database systems ". *A quarterly bulletin of the IEEE Technical Committee on Data Engineering*, 10(3):5–11, 1987.

[BGRS91] Yuri Breitbart, Dimitrios Georgakopolous, Marek Rusinkiewicz, and Abraham Silberschatz. "on rigorous transaction scheduling ". *IEEE Transactions on Software Engineering*, 17(9):954–960, 1991.

[BHG87] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *" Concurrency Control and Recovery in Database Systems"*. Addison-Wesley Publishing Company, 1987.

[BS92] Yuri Breitbart and Avi Silberschatz. "strong recoverability in multidatabase systems ". In *Proceedings of the Research Issues in Data Engineering*, pages 170–175, 1992.

[BST90] Yuri Breitbart, Avi Silberschatz, and Glenn R. Thompson. "reliable transaction management in a multidatabase system". In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 215–224, 1990.

[BST92] Yuri Breitbart, Avi Silberschatz, and Glenn R. Thompson. "transaction management issues in a failure-prone multidatabase system environment ". *The International Journal on Very Large Data Bases*, 1(1):1–39, 1992.

[DEK91] Weimin Du, Ahmed K. Elmagarmid, and Won Kim. "maintaining quasi serializability in multidatabase systems ". In *Proceedings of the Research Issues in Data Engineering*, pages 360–367, 1991.

[DELO89] Weimin Du, Ahmed K. Elmagarmid, Y. Leu, and S. Osterman. "effects of autonomy on maintaining global serializability in heterogeneous distributed database systems ". In *Proceedings of the 2nd International Conference on Data and Knowledge Systems for Manafacturing and Engineering*, pages 113–120, 1989.

[EH88] Ahmed K. Elmargarmid and A.A. Heral. "supporting updates in heterogeneous distributed database systems". In *IEEE Proceedings of the 4th International Conference on Data Engineering*, pages 564–569, 1988.

[GRS93] Dimitrios Georgakopoulos, Marek Rusinkiwicz, and Amit P. Sheth. "using tickets to enforce the serializability of multidatabase transactions". *IEEE Transactions on Knowledge and Data Engineering*, 6(1):166–180, 1993.

[LP97] Kyuwoong Lee and Seog Park. *" Chapter 7 : Optimistic Concurrency Control for Maintaining the Global Integrity Constraints in MDBSs, IFIP TC11 WG11.5 Integrity and Internal Control in Information Systems, Volume 1"*. Chapman & Hall, 1997.

[MRKS91] Sharad Mehrotra, Rajeev Rastogi, Henry F. Korth, and Abraham Silberschatz. "non-serializable execution in heterogeneous distributed database systems". In *Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems*, pages 245–252, 1991.

[MRKS92] Sharad Mehrotra, Rajeev Rastogi, Henry F. Korth, and Avi Silberschatz. "relaxing serializability in multidatabase systems ". In *Proceedings of the Research Issues in Data Engineering*, pages 205–212, 1992.

[WV90] A. Wolski and J. Veijalainen. "2pc agent method : Achieving serializability in presence of failures in a heterogeneous multidatabase ". In *Proceedings of PARBASE-90 Conference*, pages 268–287, 1990.