1993

A View-Based Approach to Relaxing Global Serializability in Multidatabase Systems

Aidong Zhang

Evaggelia Pitoura

Bharat K. Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
93-082

A VIEW-BASED APPROACH TO RELAXING GLOBAL SERIALIZABILITY IN MULTIDATABASE SYSTEMS

Aidong Zhang
Evaggelia Pitoura
Bharat Bhargava

CSD TR-93-082
December 1993
(Revised March 1994)

A View-Based Approach to Relaxing Global Serializability in Multidatabase Systems

Aidong Zhang, Evaggelia Pitoura, and Bharat K Bhargava
Department of Computer Science
Purdue University
West Lafayette, IN 47907 USA

Abstract

In this paper, we propose a new approach to ensuring the correctness of non-serializable executions. The approach is based on relating transaction views of the database to the integrity constraints of the system. Drawing upon this approach, we develop a new correctness criterion for multidatabase concurrency control. This criterion, called view-based two-level serializability, relaxes global serializability in multidatabase systems while respecting the autonomy of local database systems. No additional restrictions other than serializability are imposed on local database systems.

1 Introduction

A multidatabase system (MDBS) is a higher-level confederation of a number of pre-existing autonomous and possibly heterogeneous database systems. The purpose of a multidatabase system is to allow uniform access to the information stored in these component systems. In this environment, transaction management is performed at two levels: at a local level by the pre-existing transaction managers (LTMs) of the local databases, and at a global level by the global transaction manager (GTM). There are two types of transactions:

- local transactions, which access data at one site only and which are submitted to the appropriate LTM, outside the control of the GTM; and
- global transactions, which are submitted to the GTM, where they are parsed into a number of global subtransactions, each of which accesses data stored in a single local

database. These subtransactions are then submitted for execution to the appropriate LTM. Global subtransactions are viewed by an LTM as ordinary local transactions.

Local transaction managers are responsible for the correct execution of all transactions executed at their local sites. The global transaction manager thus retains no control over global transactions after their submission to the LTMs and can make no assumptions about their execution. The autonomous nature of the LTMs thus greatly complicates the problem of transaction management in a multidatabase system.

Concurrency control in MDBSs has received much attention from multidatabase researchers. In particular, maintaining global serializability in the execution of both local and global transactions has been well studied [2, 8, 3, 4, 10]. The difficulties in this regard center upon the inability of the GTM to control the serialization order of global subtransactions at local sites. As a result, even a serial execution of global transactions at the global level may not ensure that their serialization order at a local site will be consistent with their execution order at the global level. All successful attempts for ensuring global serializability require the enforcement of conflicting operations among global subtransactions at each local site [4, 5, 11]. The GTM can thus control the serialization order of global subtransactions by controlling the execution of these conflicting operations. However, enforcing conflicts may result in poor performance if most global transactions would not naturally conflict. Relaxing global serializability is thus a significant issue for multidatabase concurrency control. A non-serializable criterion, called two-level serializability (2LSR), was introduced in [6]. In general, 2LSR executions of local and global transactions do not guarantee database consistency.

In this paper, we present an approach to ensuring that 2LSR executions maintain multidatabase consistency. This approach draws upon the observation that the view of a global transaction, that is, the data it reads, can play an important role in ensuring that its execution will maintain database consistency. The underlying concepts of the view consistency and view closure of transactions are defined by relating the transaction views to the integrity constraints that are defined in the system. The impact of such global transaction views on correctness criteria leads to the formulation of the concept of *view-based two-level serializability.* This new correctness criterion, imposes certain restrictions on the view of global transactions that participate in two-level serializable executions. We shall demonstrate that the view-based two-level serializable execution of local and global transactions can maintain

multidatabase consistency in various practical multidatabase models. As this criterion is more general than serializability and the view consistency and view closure of transactions can be efficiently enforced by the system, the proposed approach can be applied to the execution of all global and local transactions.

This paper is organized as follows. Section 2 introduces the fundamental concepts underlying our formulation of view-based two-level serializability, the details of which are presented in Section 3. Section 4 compares the present work with related research, while concluding remarks are offered in Section 5.

2 Preliminaries

In this section, we shall introduce the basic terminology to be used in this paper and explore the background of the problem.

2.1 Basic Terminology

A multidatabase is the union of all data items stored at the participating local sites. We denote the set of all data items in a local site LS_i by D_i for $i = 1, ..., m$ and the set of all data items in the multidatabase by $\mathcal{D}$. Thus, $\mathcal{D} = \bigcup_{i=1}^{m} D_i$. We assume that local databases are disjoint; that is, $D_i \cap D_j = \emptyset, i \neq j$. To distinguish between data items prior to multidatabase integration, the set of data items at a local site LS_i is partitioned into *local* data items, denoted LD_i, and *global* data items, denoted GD_i, such that $LD_i \cap GD_i = \emptyset$ and $D_i = LD_i \cup GD_i$. The set of all global data items is denoted GD, $GD = \bigcup_{i=1}^{m} GD_i$.

Following the traditional approach, a *database state* is defined as a mapping of every data item to a value of its domain, and integrity constraints are formulas in predicate calculus that express relationships of data items that a database must satisfy. In an MDBS system, there are three types of integrity constraints: *local integrity constraints* are defined on local data items at a single local site; *local/global integrity constraints* are defined between local and global data items; and *global integrity constraints* are defined on global data items. The consistency of database state is then defined as follows:

Definition 1 (Database consistency) *A local database state is consistent if it preserves*

all integrity constraints that are defined at the local site. A multidatabase state is consistent if it preserves all integrity constraints defined in the MDBS environment.

In this paper, a *transaction* is a sequence of *read* and *write* operations resulting from the execution of a transaction program. Such a program is conventionally written in a high-level programming language, with assignments, loops, conditional statements, and other control structures. For the elements of a transaction, we denote the read and write operations as $r(x)$ and $w(x)$ (possibly subscripted). We shall alternatively use $r(x, v)$ (or $w(x, v)$) to denote an operation which reads (or writes) a value v from (or to) the data item x. Two operations *conflict* with each other if they access the same data item and at least one of them is a *write* operation. The execution of a transaction transfers a database from one consistent state to another.

In an MDBS environment, a *local transaction* is a transaction that accesses the data items at a single local site. A *global transaction* is a set of *global subtransactions*, within which each global subtransaction is a transaction accessing the data items at a single local site. In this paper, we assume that each global transaction has only one subtransaction at each local site. The execution of a global transaction transfers a multidatabase from one consistent state to another.

A *schedule* over a set of transactions is a partial order of all and only the operations of those transactions which orders all conflicting operations and which respects the order of operations specified by the transactions. In a MDBS environment, a *local schedule* S^{D_k} is a schedule over both local transactions and global subtransactions which are executed at the local site LS_k, and a *global schedule* S is a schedule over both local and global transactions which are executed in an MDBS. A *global subschedule* $S^{\mathcal{G}}$ is global schedule S restricted to the set $\mathcal{G}$ of global transactions in S. The correctness of a schedule is defined as follows:

Definition 2 (Schedule correctness) *A schedule is correct if it preserves all integrity constraints that are defined in the database system and each transaction in S reads only a consistent database state.*

2.2 Relaxing Global Serializability

A global schedule S is considered to be *globally serializable* if S is serializable [1] on the execution of both local and global transactions. Clearly, if local and global transactions maintain all integrity constraints defined in the MDBS environment, then a globally serializable global schedule is correct. In order to avoid the potential of poor performance which may be caused by global serializability, several researches have suggested notions of correctness based upon integrity constraints that are weaker than global serializability. Two well-known approaches appear in [3, 6]. In [6], a non-serializable criterion, termed *two-level serializability* is proposed:

Definition 3 (Two-level serializable global schedule) *A global schedule is* two-level serializable, *denoted 2LSR, if its global subschedule and local schedules are serializable.*

In [3], another non-serializable criterion, termed *quasi-serializability* (QSR), is proposed for global schedules. A global schedule is *quasi-serial* if its local schedules are serializable and there is a total order on global transactions such that, for any two global transactions T_i and T_j, if T_i precedes T_j in the total order, then all T_i's operations precede all T_j's operations in all local schedules in which both appear. A global schedule is *quasi-serializable* if it is conflict equivalent to a quasi-serial schedule.[1]

As stated in [7], the set of 2LSR global schedules is a superset of the set of QSR global schedules. It has also been observed that restricting the execution order of global subtransactions in 2LSR global schedules to ensure QSR global schedules does not significantly improve the correctness of global schedules. The following example given in [7] is illustrative:

Example 1 *Consider an MDBS consisting of two LDBSs, where data items a, b, e are at LS_1, and c is at LS_2. Let a, b, c, e be the local data items and the integrity constraints be $a > 0 \rightarrow b > 0$ and $c > 0$ and $e > 0$. The following two global transaction programs p_1, p_2 and one local transaction program p_L are submitted:*

p_1 : *if* $a > 0$ *then* $c = b$ *else* $c = 1$

p_2 : $e = c$

p_L : $a = 1$

[1] See [1] for the concept of conflict equivalence.

$if\ e > 0\ then\ b = 1$

Starting from a state $a = -1, b = -1, c = 1, e = 1$, *consider the following executions:*

$S_1 : w_L(a, 1) r_1(a, 1) r_1(b, -1) w_2(e, -1) r_L(e, -1)$,

$S_2 : w_1(c, -1) r_2(c, -1)$.

The resulting state is $a = 1, b = -1, c = -1, e = -1$, *which is inconsistent. Note that* S *is both two-level serializable and quasi-serializable.* □

Thus, we shall focus on two-level serializable global schedules and investigate conditions which must be placed on global transactions to ensure that two-level serializable global schedules will preserve multidatabase consistency.

Throughout this paper we assume that there are no local/global integrity constraints that are defined among different sites. This is a reasonable assumption, because due to local autonomy, local transactions may be unaware of those integrity constraints and thus unable to maintain them. Note that since, in the presence of local/global constraints, the execution of a local transaction itself may not maintain database consistency, a serializable global schedule would be incapable of maintaining correctness.

3 View-Based Two-Level Serializability

In this section, we present a new approach which ensures that two-level serializable global schedules will preserve multidatabase consistency. Consistency is preserved by restricting the views of global transactions in these schedules. The resulting schedules are called *view-based two-level serializable* global schedules.

3.1 Views of Transactions

Let t be a transaction. The read set of data items of t, denoted $RS(t)$, is the combination of the set of local data items read by operations in t (denoted $RL(t)$) and the set of global data items read by operations in t (denoted $RG(t)$). The write set of data items of t, denoted $WS(t)$, is the combination of the set of local data items written by operations in t (denoted $WL(t)$) and the set of global data items written by operations in t (denoted $WG(t)$).

Let c_i be an integrity constraint that is defined on a set of data items $\{d_1, ..., d_l\}$. Let

$D(c_i)$ denote the set of data items in c_i. Thus, we have $D(c_i) = \{d_1, ..., d_l\}$. Let $In(d)$ denote the set of data items which shares a common integrity constraint with data item d. Clearly, if c_i is the only integrity constraint that is defined in the database, then $In(d_1) = \{d_2, ..., d_l\}$, $In(d_2) = \{d_1, d_3, ..., d_l\}$, and $In(d_i) = \{d_1, ...d_{i-1}, d_{i+1}, ..., d_l\}$ for all $i = 3, ..., l$.

We now introduce the concepts of the closure of data items and of transactions which are closed on a given set of data items.

Definition 4 (Closure of data items) *Let $D = \{d_1, ..., d_l\}$ be a set of data items. The closure of D, denoted $cl(D)$, consists of all data items such that the following conditions are satisfied:*

- $D \subseteq cl(D)$.
- *If $d \in cl(D)$, then $In(d) \subseteq cl(D)$.*

Definition 5 (View closed Transaction) *A transaction t_i is view closed with respect to a set of data items D that it reads if, for any $d \in cl(D)$, $d \in RS(t_i)$.*

Let DS be the database state of $\mathcal{D}$. The restriction of DS to data items in $D \subseteq \mathcal{D}$ is denoted by DS^D. Let $read(t)$ denote the database state seen as a result of the read operations in t and $read(t^D)$ denote the database state of D seen as a result of the read operations in transaction t. Let $write(t)$ denote the effect on the database as a result of the write operations in t and $write(t^D)$ denote the effect on the database of D as a result of the write operations in t.

We say that the view of a transaction t is consistent if $read(t)$ is consistent. In the MDBS environment, the consistency of various views of transactions is defined as follows:

Definition 6 (Local view consistency) *A transaction t_i is* local view consistent *at LS_j if $read(t_i^{LD_j})$ is consistent.*

Definition 7 (Global view consistency) *A transaction t_i is* global view consistent *if $read(t_i^{GD})$ is consistent.*

3.2 Basic Lemmas

We shall now introduce the basic lemmas involved in the development of our theory.[2]

[2]To facilitate comparison, we shall largely adopt the system of notation used in [9].

The following lemma relates the consistency of a database state to the consistency of its subsets.

Lemma 1 *Let $C_1, ..., C_n$ be the conjunction ($\wedge$) of integrity constraints, where C_i is defined over the set of data items in $D_i \subseteq \mathcal{D}$ for all $i = 1, ..., n$ and $D_i \cap D_j = \emptyset$ for all $i \neq j$. Let $D_i' \subseteq D_i$ and DS be a database state of $\mathcal{D}$. $DS^{D_i'}$, for all $i, i = 1, ..., n$, is consistent if and only if $\bigcup_{i=1}^{n} DS^{D_i'}$ is consistent.*

Proof: See Lemma 1 in [9]. □

As pointed out in [9], if lemma 1 is to hold, it is essential for the data items over which conjuncts are defined to be disjoint.

Let $\{DS_1\}t\{DS_2\}$ denote that, when transaction t executes from a database state DS_1, it results in a database state DS_2. Without loss of generality, whenever we say $\{DS_1\}t\{DS_2\}$, we assume that it is possible for t to be executed from DS_1. The conditions required to ensure that the execution of a transaction preserves the consistency of the state of a set of data items are specified as follows [7]:

Lemma 2 *Let t be a transaction and $D \subseteq \mathcal{D}$. Let $\{DS_1\}t\{DS_2\}$ and DS_1 be the database state in which t can be executed. If $DS_1^D \cup read(t)$ is consistent, then DS_2^D is consistent.*

Proof (sketch): By the definition of consistency, there exists a consistent state DS_3 such that $DS_3^{D \cup RS(t)} = DS_1^D \cup read(t)$. Let $\{DS_3\}t\{DS_4\}$. Since t itself preserves database consistency, DS_4 is consistent. Also, since $DS_1^D = DS_3^D, DS_2^D = DS_4^D$. Hence, DS_2^D is consistent. □

We may now relate the consistency of a database state to the execution of transactions. The state associated with a transaction in a schedule is a possible state of the data items that the transaction may have seen. Let $\tau_w(D, S)$ denote the set of transactions in a schedule S that have at least one write operation on some data item in $D \subseteq \mathcal{D}$. Let S be a schedule and $D \subseteq \mathcal{D}$ such that $(S^\tau)^D$ is serializable, where $\tau_w(D, S) \subseteq \tau$. Let $t_1, ..., t_n$ be a serialization order of transactions in $(S^\tau)^D$ and DS_1 be a database state from which S starts. The state of the database before the execution of each transaction, with respect to data items in D, is defined as follows:

$$state(t_i, D, S, DS_1) = \begin{cases} DS_1^D, & \text{if } i = 1 \\ state(t_{i-1}, D, S, DS_1)^{D-WS(t_{i-1}^D)} \cup write(t_{i-1}^D), & \text{if } i > 1 \end{cases}$$

Note that $read(t_i^D) \subseteq state(t_i, D, S, DS)$. Since the execution of both local and global transactions must transfer the database from one consistent multidatabase state to another consistent state, it is essential for each transaction in a global schedule to see a consistent state. We now use Lemma 2 to develop the conditions under which each transaction in a schedule reads a database state that is consistent with respect to a set of data items.

Lemma 3 *Let $C_1, ..., C_m$ be the conjunction ($\wedge$) of integrity constraints, where C_k is defined over the set of data items in $D_k \subseteq \mathcal{D}$ for all $k = 1, ..., m$ and $D_i \cap D_j = \emptyset$ for all $i \neq j$. Let S be a schedule and $\{DS_1\}S\{DS_2\}$. If, for any $k = 1, ..., m$,*

- *$(S^\tau)^{D_k}$ is serializable with serialization order $t_1, ..., t_n$, where $\tau_w(D_k, S) \subseteq \tau$,*
- *$read(t_i^{\mathcal{D}-D_k})$ is consistent for all $t, t \in \tau_w(D_k, S)$, and*
- *$DS_1^{D_k}$ is consistent,*

then $state(t_i, D_k, S, DS_1)$ is consistent for all $t_i, i = 1, ..., n$.

Proof (sketch): The proof proceeds by induction on the number n of transactions.

Basis: ($n = 1$) Clearly, $state(t_1, D_k, S, DS_1) = DS_1^{D_k}$, which is consistent.

Induction: Suppose $state(t_l, D_k, S, DS_1)$ is consistent. We need to show that $state(t_{l+1}, D, S, DS_1)$ is consistent. Consider two cases:

(1) $t_l \notin \tau_w(D_k, S)$. Thus, $state(t_{l+1}, D, S, DS_1) = state(t_l, D_k, S, DS_1)$. By induction hypothesis, $state(t_l, D_k, S, DS_1)$ is consistent. Hence, $state(t_{l+1}, D, S, DS_1)$ is consistent.

(2) $t_l \in \tau_w(D_k, S)$. Since $state(t_l, D_k, S, DS_1)$ and $read(t_l^{\mathcal{D}-D_k})$ are consistent, by Lemma 1, we see that $state(t_l, D_k, S, DS_1) \cup read(t_l)$ is consistent. By Lemma 2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. □

From Lemma 3, we see that transaction views play an important role in ensuring that all transactions in a global schedule see a consistent state. Below, we shall investigate the effect of the local and global views of global transactions on the maintenance of multidatabase consistency in two practical MDBS models.

3.3 The Global Read-Write (G_{rw}) Model

The G_{rw} model is defined as satisfying the following conditions:

- local transactions read and write only local data items, and
- global transactions read and write both local and global data items.

This model is applicable to an MDBS environment, where the originally independent constituent databases may be viewed as the local data items, accessed by the original local transactions. Global data items are then added by storing new data items in these databases.

We now apply Lemma 3 to a specific G_{rw} model and show that, when global transactions are local view consistent, global transactions read consistent data:

Lemma 4 *Let S be a 2LSR schedule in the G_{rw} model with no integrity constraints present between local and global data items. Let DS_1 be a consistent database state from which S starts. If all global transactions in S are local view consistent, then, for all global transactions t_i in S, $read(t_i)$ is consistent.*

Proof (sketch): Since no integrity constraints are present between local and global data items, the integrity constraints can be viewed as $C_1, ..., C_{m+1}$. Here, C_i for $i = 1, ..., m$ are the conjuncts ($\wedge$) of integrity constraints that are defined over the sets of data items in LD_i for $i = 1, ...m$, respectively, and C_{m+1} is the conjunct of integrity constraints that are defined over the set of data items in GD. Following the G_{rw} model, we have $LD_i \cap LD_j = \emptyset$ for $i \neq j$, and $LD_i \cap GD = \emptyset$. Since, in the G_{rw} model, only the set $\mathcal{G}$ of global transactions access GD and $S^{\mathcal{G}}$ is serializable, $(S^{\mathcal{G}})^{GD}$ is serializable. Let $t_1, ..., t_n$ be the serialization order of the global transactions in $(S^{\mathcal{G}})^{GD}$. Since the global transactions are local view consistent, $read(t_i^{D-GD})$ is consistent for all $i = 1, ..., n$. By Lemma 3, $state(t_i, GD, S, DS_1)$ is consistent for all $i = 1, ..., n$. Since $read(t_i^{GD}) \subseteq state(t_i, GD, S, DS_1)$, $read(t_i^{GD})$ is consistent. Thus, by Lemma 1, $read(t_i)$ is consistent for all $i = 1, ..., n$. □

Following Lemma 4, for those global transactions in a 2LSR global schedule in the G_{rw} model with no integrity constraints present between local and global data items, local view consistency implies global view consistency.

It follows from Lemmas 3 and 4 that, given that global transactions are local view consistent, local transactions read consistent data:

Corollary 1 *Let S be a 2LSR schedule in the G_{rw} model with no integrity constraints present between local and global data items. Let DS_1 be a consistent database state from which S starts. If all global transactions in S are local view consistent, then, for all local transactions t_i in S, $read(t_i)$ is consistent.*

Proof (sketch): Since S^{D_k} is serializable, S^{LD_k} is serializable for all $k = 1, ..., m$. Let $t_1, ..., t_n$ be the serialization order of the transactions in S^{LD_k}. By Lemma 4, $read(t_i^{D-LD_k})$ is consistent for all $i = 1, ..., n$. By Lemma 3, $state(t_i, LD_k, S, DS_1)$ is consistent for all $i = 1, ..., n$. For any local transaction t_i in S^{LD_k}, since $read(t_i) \subseteq state(t_i, LD_k, S, DS_1)$, $read(t_i)$ is consistent. □

We now are able to demonstrate that, if global transactions are local view consistent, then 2LSR global schedules preserve multidatabase consistency.

Theorem 1 *Let S be a 2LSR schedule in the G_{rw} model with no integrity constraints present between local and global data items. If all global transactions in S are local view consistent, then S is correct.*

Proof (sketch): Let DS_1 be a consistent multidatabase state and $\{DS_1\}S\{DS_2\}$. We need to show that all transactions in S read consistent data and that DS_2 is consistent. By Lemma 4 and Corollary 1, for all transactions t_i in S, $read(t_i)$ is consistent. Now, let S^{LD_k} be serializable with serialization order $t_1, ..., t_n$. From Lemma 3, $state(t_n, LD_k, S, DS)$ is consistent. Hence, there exists a consistent database state DS_3 such that $DS_3^{LD_k} = state(t_n, LD_k, S, DS)$ and $DS_3^{RS(t_n)} = read(t_n)$. Thus, t_n can be executed in DS_3. Let $\{DS_3\}t_n\{DS_4\}$. Since $DS_3^{LD_k} \cup read(t_n)$ is consistent, by Lemma 2, $DS_4^{LD_k}$ is consistent. Since $DS_2^{LD_k} = DS_4^{LD_k}$, $DS_2^{LD_k}$ is consistent. Hence, for all $i, i = 1, ..., m$, $DS_2^{LD_i}$ is consistent. Similarly, DS_2^{GD} is consistent. By Lemma 1, DS_2 is consistent. Hence, S is correct. □

In Example 1, all data items are local and no integrity constraints exist between different local sites. However, since both global transactions in the given global schedule have inconsistent local views, Theorem 1 cannot be applied. If we require that $r_1(a)r_1(b)$ and $r_2(c)$ be consistent, then both $w_1(c)$ and $w_2(e)$ would be consistent. As a result, the local transaction would not read inconsistent data, thus resulting in a consistent local database state.

3.4 The Global Read-Write and Local Read ($G_{rw}L_r$) Model

The $G_{rw}L_r$ model is defined as satisfying the following conditions:

- local transactions read and write local data items and also read global data items, and
- global transactions read and write global and local data items.

The $G_{rw}L_r$ model extends the G_{rw} model by allowing new local transactions to read the new global data items.

The results presented in the previous subsection cannot be directly applied to the $G_{rw}L_r$ model. The following example is illustrative [6]:

Example 2 *Consider an MDBS consisting of two LDBSs, where data items b, c, e are at LS_1, and a is at LS_2. Let e be the local data item, a, b, c be global data items, and the integrity constraints be $a > 0 \rightarrow c > 0$ and $b > 0 \rightarrow c > 0$ and $e > 0$. The following two global transaction programs p_1, p_2 and one local transaction program p_L are submitted:*

p_1 : $b = 1$

$\quad$ *if* $a \leq 0$ *then* $c = 1$

p_2 : $a = 1$

$\quad$ $c = 1$

p_L : *if* $b > 0$ *then* $e = c$ *else* $e = 1$

Starting from a state $a = -1, b = -1, c = -1, e = 1$, consider the following executions:

$S_1 : w_1(b, 1) r_L(b, 1) r_L(c, -1) w_2(c, 1) w_L(e, -1)$,

$S_2 : w_2(a, 1) r_1(a, 1)$.

The resulting state is $a = 1, b = 1, c = 1, e = -1$, which is inconsistent. □

In Example 2, the local transaction reads inconsistent global data. The problem here is that, although $read(t_1^{GD})$ is consistent in S^{GD}, $t_1^{D_1}$ is executed in a different local database state, in which some global integrity constraints are actually not satisfied. Note that Lemma 4 still holds in this context; however, Corollary 1 does not hold.

The following lemma shows that, if a global transaction is view closed with respect to the global data items it reads, then the union of the local database state it sees and the data it reads is consistent.

Lemma 5 *Let S be a 2LSR schedule in the $G_{\tau w}L_{\tau}$ model with no integrity constraints present between local and global data items. Let $\{DS_1\}S\{DS_2\}$ and DS_1 be consistent. If all global transaction in S are local view consistent and view closed with respect to global data items they read and, for any global transaction t_i, $state(t_i, D_k, S, DS_1)$ is consistent, then $state(t_i, D_k, S, DS_1) \cup read(t_i)$ is consistent.*

Proof (sketch): Since only the set $\mathcal{G}$ of global transactions write on global data items, $\tau_w(GD, S) \subseteq \mathcal{G}$. Since $S^{\mathcal{G}}$ is serializable, $(S^{\mathcal{G}})^{GD}$ is serializable. By Lemmas 3 and 4, $state(t_i, GD, S, DS_1)$ and $read(t_i)$ are consistent. Suppose now that $state(t_i, D_k, S, DS_1) \cup read(t_i)$ is not consistent. There must then be an integrity constraint c between D_k and GD which is not satisfied. Let $D(c)$ denote the data items in c. We have $D(c) \cap RG(t_i) \neq \emptyset$. Since t_i is view closed with respect to global data items, $D(c) \subseteq RG(t_i)$. Thus, $read(t_i)$ is not consistent, which is contradictory to the previous result. Hence, $state(t_i, D_k, S, DS_1) \cup read(t_i)$ is consistent. □

Lemma 6 *Let S be a 2LSR schedule in the $G_{\tau w}L_{\tau}$ model with no integrity constraints present between local and global data items. Let S^{D_k} is serializable with serialization order $t_1, ..., t_n$. Let $\{DS_1\}S\{DS_2\}$ and DS_1 be consistent. If all global transactions in S are local view consistent and view closed with respect to global data items they read, then $state(t_i, D_k, S, DS_1)$ is consistent for all $t_i, i = 1, ..., n$.*

Proof (sketch): The proof proceeds by induction on the number n of transactions.

Basis: $(n = 1)$ Clearly, $state(t_1, D_k, S, DS_1) = DS_1^{D_k}$, which is consistent.

Induction: Suppose $state(t_l, D_k, S, DS_1)$ is consistent. We need to show that $state(t_{l+1}, D_k, S, DS_1)$ is consistent. If t_l is a local transaction, then $read(t_l) \subseteq state(t_l, D_k, S, DS_1)$. Thus, by Lemma 2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. Now we consider t_l to be a global subtransaction. Following Lemma 5, $state(t_l, D_k, S, DS_1) \cup read(t_l)$ is consistent. Again, by Lemma 2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. □

The following theorem based upon Lemmas 5 and 6 illustrates the conditions under which 2LSR schedules preserve database consistency in the $G_{\tau w}L_{\tau}$ model.

Theorem 2 *Let S be a 2LSR schedule in the $G_{\tau w}L_{\tau}$ model with no integrity constraints present between local and global data items. If all global transactions in S are both local and global view closed, then local view consistency implies that S is correct.*

Proof (sketch): Let DS_1 be a consistent multidatabase state and $\{DS_1\}S\{DS_2\}$. We need to show that all transactions in S read consistent data and that DS_2 is consistent. Since only the set $\mathcal{G}$ of global transactions write on global data items, $\tau_w(GD, S) \subseteq \mathcal{G}$. Since $S^{\mathcal{G}}$ is serializable, $(S^{\mathcal{G}})^{GD}$ is serializable. By Lemma 4, all global transactions read consistent data. Following Lemma 6, all local transactions also read consistent data. Thus, for all transaction t_i in S, $read(t_i)$ is consistent. Now, let S^{D_k} be serializable with serialization order $t_1, ..., t_n$. Since, from Lemma 6, $state(t_n, D_k, S, DS)$ is consistent, $state(t_n, LD_k, S, DS)$ is then consistent. Hence, there exists a consistent database state DS_3 such that $DS_3^{LD_k} = state(t_n, LD_k, S, DS)$ and $DS_3^{RS(t_n)} = read(t_n)$. Thus, t_n can be executed in DS_3. Let $\{DS_3\}t_n\{DS_4\}$. Since $DS_3^{LD_k} \cup read(t_n)$ is consistent, by Lemma 2, $DS_4^{LD_k}$ is consistent. Since $DS_2^{LD_k} = DS_4^{LD_k}$, $DS_2^{LD_k}$ is consistent. Hence, for all $i, i = 1, ..., m$, $DS_2^{LD_i}$ is consistent. Similarly, DS_2^{GD} is consistent. By Lemma 1, DS_2 is consistent. Hence, S is correct. □

In Example 2, no integrity constraints are defined between local and global data items. However, since $r_1(a)$ in global transaction t_1 in the given global schedule is not global view closed, Theorem 2 cannot be applied. Suppose that now we require the view of t_1 to be closed as $r_1(a)r_1(c)r_1(b)$ and t_1 to be serialized after t_2 in S_G. In this example, t_1 and t_2 do not read and write local data, and each global transaction would therefore transfer global data items from one consistent state to another. Hence, the local transaction L reads consistent global data and results in a consistent local database state.

4 Relationship to Other Research

In the previous section we advanced certain prerequisites to the correctness of 2LSR global schedules. In particular, we have shown that 2LSR global schedules are correct in the G_{rw} model if global transactions possess a consistent local view. We have also shown that 2LSR global schedules are correct in the $G_{rw}L_r$ model if global transactions possess a consistent local view and a closed global view. In both cases, no additional restrictions other than serializability need be imposed on LDBSs. In this section, these conditions will be compared with those advanced in the literature.

The correctness of 2LSR global schedules in the G_{rw} model has been examined in [6]. To avoid inconsistencies, both local and global transaction programs are required to be fixed-structured. A transaction program is *fixed-structured* if its execution from every database

state results in transactions with a common structure. The correctness of 2LSR global schedules in the $G_{rw}L_r$ model has also been examined in [6]. To avoid inconsistencies, global transaction programs must possess no value dependencies among their global subtransactions. A global subtransaction t_j is *value dependent* on a set of global subtransactions $t_1, ..., t_{j-1}$ if the execution of one or more operations in t_j is determined by the values read by $t_1, ..., t_{j-1}$.

It is illuminating to compare the range of acceptable schedules generated by the present work with those encompassed by the above method. Let ST_2LSR denote the set of 2LSR global schedules in which all transactions are fixed-structured; ND_2LSR denote the set of 2LSR global schedules with no value dependencies permitted in global transactions; LV_2LSR denote the set of 2LSR global schedules in which the local views of global transactions are consistent; and LG_2LSR denote the set of 2LSR global schedules in which the local views of global transactions are consistent and the global view of global transactions are closed.

Within the G_{rw} model, since ST_2LSR global schedules are correct, the fact that both local and global transactions are fixed-structured implies that their retrievals from local sites will be consistent. However, the possession of consistent local views by global transactions does not imply that both local and global transactions are fixed-structured. Thus, LV_2LSR is a superset of ST_2LSR. Within the $G_{rw}L_r$ model, the fact that a global transaction has no value dependencies does not imply that its retrieval of global data items is closed; nor does the converse hold true. Thus, there is no inclusive relationship between ND_2LSR and LG_2LSR.

We now compare further the above conditions in terms of their applicability in the multidatabase environment. As pointed out in [6] it may be impractical to assume the presence of fixed structured programs, since local transaction programs are pre-existing and may not satisfy these restrictions. Similarly, the prohibition of value dependencies is excessively restrictive, as many applications involve data transfer among different local database sites, resulting in value dependencies among the subtransactions of a global transaction. In contrast, our approach is more practical, since it affects only global transactions and the testing of local view consistency as well as the specifications of global view closures in global transactions can be easily implemented.

[9] presented additional findings relevant to the present research. That work presented

a non-serializable criterion, termed *predicatewise serializability* (PWSR), to be applied in a database environment in which the integrity constraints can be grouped into $C_1 \wedge ... \wedge C_l$, where C_i is defined over a set of data items $d_i \subseteq D$ and $d_i \cap d_j = \emptyset, i \neq j$. A schedule is said to be PWSR if, for all i, $i = 1, ..., l$, S^{d_i} is serializable. That research demonstrated that a PWSR schedule S is correct, either if all transaction programs have a fixed-structure or if S is a delayed read schedule. A schedule S is *delayed read* if each transaction T_i in S cannot read a data item written by transaction T_j until the completion of all T_j's operations. This theory may be applied to an MDBS environment in which all local schedules are serializable (termed *local serializability*) and either both local and global transactions are fixed-structures or all local schedules are delayed read. Clearly, the present work has advantages over the application of PWSR in the MDBS environment, since PWSR is applicable only if local transactions have a fixed structure or local schedules are delayed-read.

5 Conclusions

In this paper, we have proposed a new view-based approach to ensuring the correctness of non-serializable schedules. This approach rests upon the concepts of the view consistency and view closure of transactions, through which data items read by transactions are related to the integrity constraints that are defined on these data items. The benefits from this approach become clear through its application to the formulation of correct non-serializable global schedules in multidatabase systems.

Global serializability has been recognized as excessively restrictive for the MDBS environment, encouraging the development of more relaxed correctness criteria. Drawing upon view consistency and view closures, we have proposed a new criterion, called view-based two level serializability. View-based two-level serializable schedules were shown to preserve multidatabase consistency. Furthermore, this criterion respects local autonomy, since no restrictions other than serializability need be imposed on local schedules. Finally, as the concepts of view consistency and view closure rest solely upon the structural properties of the integrity constraints rather than their semantics, such restrictions can be enforced systematically.

In summary, the main contributions of this paper are as follows:

- The introduction of the concept of view consistency and view closure of transactions. We believe that the relation of transaction views to integrity constraints provides an innovative approach to maintaining database consistency.

- The development of a new correctness criterion for multidatabase systems. The new criterion, termed view-based two-level serializability, uses the concept of view consistency and view closure, to specify conditions that permit 2LSR global schedules to ensure multidatabase consistency.

In future studies, this view-based approach will be applied to other MDBS models. Future research will also explore efficient mechanisms for calculating view closures.

Acknowledgements

We would like to acknowledge Rachel Ramadhyani for her inputs toward the presentation of this paper.

References

[1] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Databases Systems.* Addison-Wesley Publishing Co., 1987.

[2] Y. Breitbart and A. Silberschatz. Multidatabase Update Issues. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 135–142, June 1988.

[3] W. Du and A. Elmagarmid. Quasi Serializability: a Correctness Criterion for Global Concurrency Control in InterBase. In *Proceedings of the 15th International Conference on Very Large Data Bases*, pages 347–355, Amsterdam, The Netherlands, Aug. 1989.

[4] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. On Serializability of Multidatabase Transactions Through Forced Local Conflicts. In *Proceedings of the 7th Intl. Conf. on Data Engineering*, pages 314–323, Kobe, Japan, Apr. 1991.

[5] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. The Concurrency Control Problem in Multidatabases: Characteristics and Solutions. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 288–297, 1992.

[6] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. Maintaining database consistency in heterogenous distributed database systems. Technical Report TR-91-04, University of Texas at Austin Department of Computer Science, 1991.

[7] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. Non-serializable executions in heterogenous distributed database systems. In *Proceedings of 1st International Conference on Parallel and Distributed Information Systems*, pages 245–252, Dec. 1991.

[8] C. Pu. Superdatabases for Composition of Heterogeneous Databases. In *Proceedings of the International Conference on Data Engineering*, pages 548–555, Feb. 1988.

[9] R. Rastogi, S. Mehrotra, Y. Breitbart, H. F. Korth, and A. Silberschatz. On correctness of non-serializable executions. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 97–108, May 1993.

[10] J. Veijalainen and A. Wolski. Prepare and commit certification for decentralized transaction management in rigorous heterogeneous multidatabases. In *Proc. Int'l Conf. On Data Engineering*, 1992.

[11] A. Zhang and A. K. Elmagarmid. A theory of global concurrency control in multidatabase systems. *The VLDB Journal*, 2(3):331–359, July 1993.