1994

Maintaining Consistency in Multidatabase Systems: A Comprehensive Study

Aidong Zhang

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

Report Number:
94-005

MAINTAINING CONSISTENCY IN MULTIDATABASE SYSTEMS; A COMPREHENSIVE STUDY

Aidong Zhang
Bharat Bhargava

Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD TR-94-005
January 1994

Maintaining Consistency in Multidatabase Systems: A Comprehensive Study

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

Abstract

This paper develops a fundamental understanding of multidatabase concurrency control, which reveals the principles of multidatabase concurrency control to accommodate the two-level hierarchy of globally and locally independent transaction managements. Local extensibility is proposed as a unifying principle in the combination of global and local transaction managements. Under this general principle, the key issues of ensuring the isolation and semantic atomicity of global transactions are addressed. Correctness criteria are formulated for various situations and recoverable global transactions are constructed. Our discussions are conducted upon the only basic assumptions of serializability and recoverability on local database systems. The preservation of the necessary balance between the demands of global transaction management and local autonomy provide us with an enhanced theoretical understanding of the limitations of global transaction management in multidatabase systems.

Key Words: Multidatabases, transaction management, concurrency control, local extensibility, compensating serializability, recoverability

1 Introduction

This paper investigates those problems that arise in multidatabase systems (MDBSs) at the level of transaction management. The role of a transaction manager is the preservation of the atomicity, isolation, and durability [Gra81, HR83, ÖV91] of transactions. In a multidatabase model, transaction management is handled at both the global and local levels. A global transaction manager (GTM) is superimposed upon a set of local autonomous database systems (LDBSs). Global transactions are submitted to the global transaction manager, where they are parsed into a set of global subtransactions to be individually submitted to local transaction management systems. At the same time, local transactions are directly submitted to the local transaction management systems. Each local transaction management system preserves the atomicity, isolation, and durability of both local and global subtransactions at its site. It is left to the global transaction manager to maintain the atomicity and isolation of global transactions.

The overriding concern of any MDBS is the preservation of local autonomy. Aspects of local autonomy such as design, execution, and control have been studied in [Lit86, GMK88, BS88, Pu88, Vei90], and their effect on multidatabase systems is discussed in [DEK90]. By definition, a multidatabase system may not have full control over its component database systems, and it must be structured to accommodate the heterogeneity of local database systems. The autonomy of its component databases distinguishes multidatabase systems from traditional distributed database systems. Therefore, many of the early techniques developed for distributed database systems are not applicable to multidatabase systems, necessitating the formulation of new principles and protocols.

The goal of concurrency control is to ensure that transactions behave as if they are executed in isolation. The most popular correctness criterion for concurrency control is serializability [BHG87][1]. The difficulty of maintaining serializability in multidatabase systems has been made evident in the recent literature [BS88, Pu88, DE89, GRS91, VW92]. To preserve the isolation of global transactions without violation of local autonomy, a global concurrency controller or scheduler must ensures the correct execution of global transactions while allowing such executions to interleave with the globally uncontrolled execution of local transactions at local sites (LSs). Since global subtransactions are received by local transaction management systems and treated there as local transactions, the global concurrency controller must formulate its correctness criterion in a manner which is consistent with the local level.

In this paper, we shall study the principles of multidatabase transaction management in a scenario in which the local database systems are required only to ensure serializability and recoverability [BHG87]. In particular, we shall advance a unifying principle that guides multidatabase transaction management without placing additional restrictions on local database systems. This

[1] In this paper, serializability refers to conflict serializability.

is accomplished by defining properties pertaining to the execution of global transactions such that they still hold when interleaved with the globally uncontrolled execution of local transactions. We then discuss in detail the enforcement of this principle. The approach proposed here exploits the potential of a combination of global and local transaction management in the two-level hierarchical multidatabase architecture while preserving local autonomy. The preservation of the necessary balance between the demands of global transaction management and local autonomy provide us with an enhanced theoretical understanding of the limitations of global transaction management.

2 The MDBS Model and Terminology

In this section, we shall provide a precise definition of the system under consideration and introduce basic notation and terminology.

2.1 The Hierarchical System Model

A multidatabase is the union of all data items stored at the participating local sites. An MDBS consists of a set of $\{LDBS_i$, for $1 \leq i \leq m\}$, where each $LDBS_i$ is an autonomous database management system on a set of data items D_i at the local site LS_i; a set of servers associated with each LDBS; and a global transaction manager (GTM), which is superimposed on the LDBSs and servers. 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, $GD = \bigcup_{i=1}^{m} GD_i$. Figure 1 illustrates this model.

We assume that the GTM submits global transaction operations to the LDBSs through the servers, which act as the interface between the GTM and the LDBSs. The operations belonging to one global subtransaction are then submitted to an individual LDBS by the server as a single transaction. We also assume that the completion of these submitted operations is acknowledged by the LDBSs to the GTM through the servers. The GTM can thus control the execution order of global transactions by controlling their submission order.

We consider that each LDBS can deal with those failures which may occur at its local site, such as local transaction and global subtransaction failures, as well as with system and media failures [BHG87]. The GTM must have the ability to respond to additional failures, such as global transaction, server, site, and communication failures [BST92]. In this paper, we shall consider only global transaction failures. The failure of a global transaction is indicated by an aborting of its subtransaction.

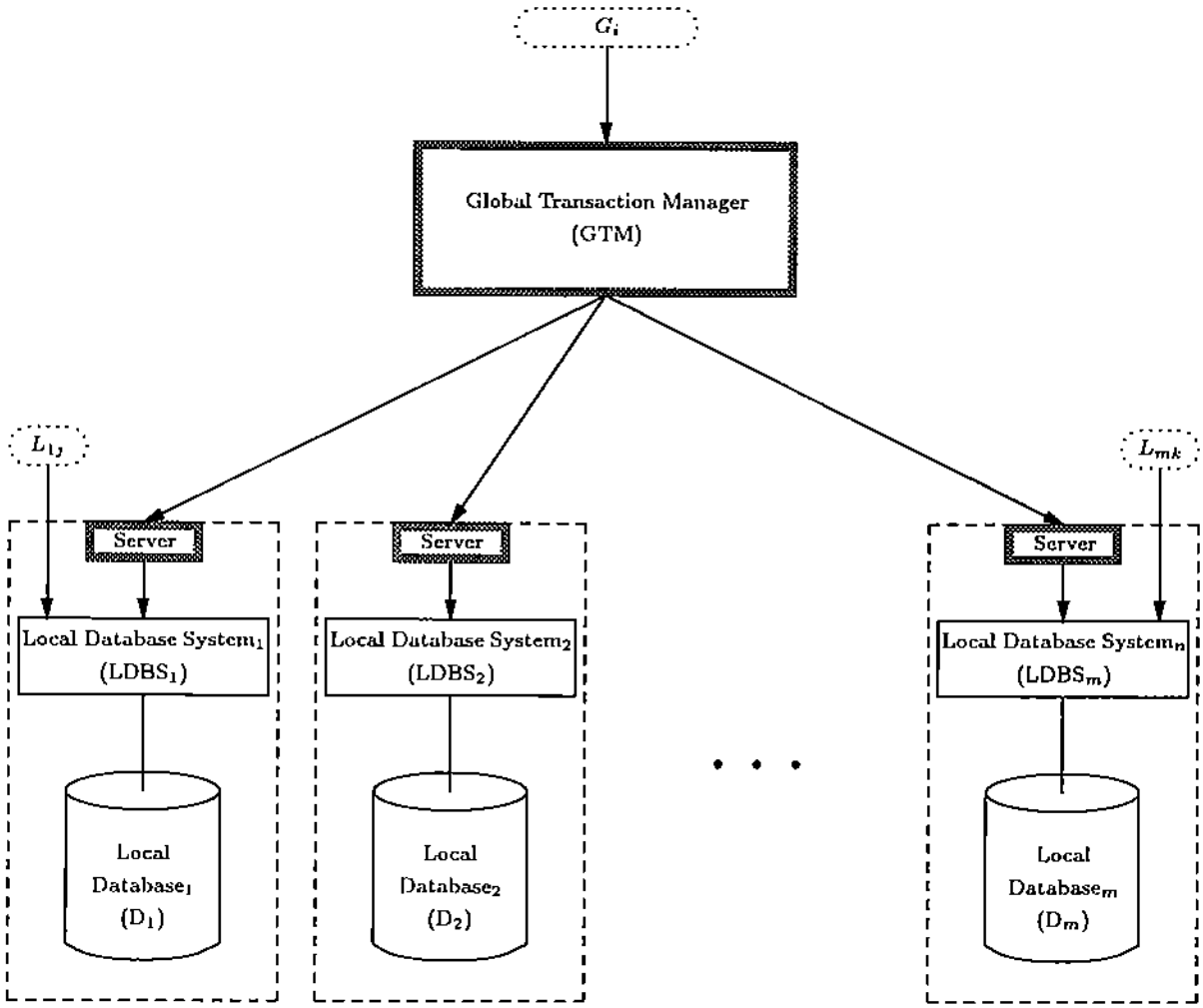

Figure 1: Two-level conceptual multidatabase architecture.

As a necessary assumption of this paper, we presume that the concurrency control and failure recovery mechanisms of LDBSs ensure serializability and recoverability. However, no restriction is imposed on these mechanisms.

2.2 Basic Terminology

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 2.1 (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. A more formal definition of a schedule can also be found in [BHG87, Had88]. A local schedule S_k is a schedule over both local transactions and global subtransactions which are executed at the local site LS_k. A global schedule S is the combination of all local schedules, while a global subschedule $S_\mathcal{G}$ is S restricted to the set $\mathcal{G}$ of global transactions in S. We denote $o_1 \prec_s o_2$ if operation o_1 is executed before operation o_2 in schedule s. We denote $T_1 \prec^s_{sr} T_2$ if T_1 precedes T_2 in the serialization order of s. The correctness of a schedule is defined as follows:

Definition 2.2 (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.*

Following [BHG87, Had88], we assume the availability of four basic transaction operations: $r(x), w(x), c$, and a, where c and a are *commit* and *abort* termination operations and $r(x)$ and $w(x)$ are *read* and *write* operations in a local database. Two operations *conflict* with each other if they access the same data item and at least one of them is a *write* operation. A transaction is a partial order of read, write, commit, and abort operations which must specify the order of conflicting operations and which contains exactly one termination operation as the maximum (last) element in the partial order. A more formal definition of a transaction can be found in [BHG87, Had88]. OP_T denotes the set of operations contained in transaction T. Two local transactions T_i and T_j

conflict, denoted $T_i \overset{c}{\sim} T_j$, if there exist conflicting operations o_i and o_j such that $o_i \in OP_{T_i}$ and $o_j \in OP_{T_j}$.

The operations in different subtransactions of a global transaction are also partially ordered according to their value dependencies, which is defined as follows:

Definition 2.3 (Value dependency) *Let global transaction* $G_i = \{G_{i1}, G_{i2}, \cdots, G_{im}\}$. G_{ij_t} *is* value dependent *on* $G_{ij_1}, ..., G_{ij_{t-1}}$ $(1 \leq j_1, ..., j_t \leq m)$, *denoted* $G_{ij_1} \rightarrow_v G_{ij_t}$, $G_{ij_2} \rightarrow_v G_{ij_t}$, ..., $G_{ij_{t-1}} \rightarrow_v G_{ij_t}$, *if there exist* $w_{ij_t}(x_{ij_t}) \in OP_{G_{ij_t}}, r_{ij_{t-1}}(x_{ij_{t-1}}) \in OP_{G_{ij_{t-1}}}, ..., r_{ij_{t-1}}(x_{ij_{t-1}}) \in OP_{G_{ij_{t-1}}}$ *such that* $x_{ij_t} = f(x_{ij_{t-1}}, ..., x_{ij_{t-1}})$ *for some function* f.

That is, the execution of an write operation in G_{ij_t} is determined by the values read by $G_{ij_1}, ..., G_{ij_{t-1}}$. We assume that value-dependencies are the only relationships defined among the global subtransactions of each global transaction.

3 A General Principle – Local Extensibility

In this section, we shall advance a general principle that unifies global and local transaction management so as to maintain the correct execution of global and local transactions in an MDBS environment without violation of local autonomy. Due to local autonomy, the global transaction manager cannot predict local interactions without placing restrictions on local sites. It is therefore appropriate to instead place restrictions on the global level to accommodate the local autonomous environments.

Let S be a global schedule and S_G be its global subschedule. At the global level, the GTM can control the submission of global transactions to local sites and, consequently, the execution order of global transactions in S. However, it must also maintain the correct execution of global transactions in S, with S containing a mixture of operations from both global and local transactions. At the local level, each LDBS freely executes both local transactions and global subtransactions as long as local correctness criteria are preserved. Since the global schedule S is the combination of all local schedules, the correct execution of local transactions is already guaranteed at local sites. Thus, the main focus of multidatabase transaction management must be to maintain the correct execution of global transactions in respect to the globally uncontrolled local interactions at local sites.

Given any properties of a global subschedule enforced by the GTM, the interaction of local transactions may in some manner change these properties, presenting obstacles to the investment of global schedules with novel properties. A well-known problem of the effects of local indirect conflicts on ensuring global serializability on global schedules illustrates the situation.

Let O be a total order on transactions. We say that an order O' is consistent with O if O' is a subsequence of O. We assume that a global subtransaction takes the name as the global transaction

to which it belongs as its order symbol in the serialization order. The following theorem identified in [MRB+92] states that a global schedule S is serializable if and only if each local restriction of S is serializable and there exists a total order O on the global transactions in S such that, in each local schedule of S, the serialization order of its global subtransactions is consistent with O.

Theorem 3.1 (Global serialization theorem) *If S is a global schedule, then S is serializable if and only if all S_k $(k = 1, ..., m)$ are serializable and there exists a total order O on global transactions in S such that for each local site $LS_k (1 \leq k \leq m)$, the serialization order of global subtransactions in S_k is consistent with O.*

However, even a serial execution of global subtransactions at a local site may not ensure that their serialization order will be consistent with their execution order.

Example 3.1 *Consider an MDBS that has data item a in LS_1 and b,c in LS_2. The following global transactions are submitted:*

$G_1 : w_{G_{11}}(a) r_{G_{12}}(b), \quad G_2 : r_{G_{21}}(a) w_{G_{22}}(c)$

Let L_{21} be a local transaction submitted at local site LS_2:

L_{21}: $w_{L_{21}}(b) w_{L_{21}}(c)$.

Let S_1 and S_2 be local schedules:

$S_1 : w_{G_{11}}(a) r_{G_{21}}(a)$

$S_2 : w_{L_{21}}(b) r_{G_{12}}(b) w_{G_{22}}(c) w_{L_{21}}(c)$

and $S = \{S_1, S_2\}$. Though the execution orders of global transactions at both local sites are $G_1 \rightarrow G_2$, the serialization order of S_2 is $G_{22} \rightarrow L_{21} \rightarrow G_{12}$. The serialization order of global subtransactions at local site LS_2 is not consistent with their execution order; this arises from the indirect conflict of G_{22} with G_{12} (since $w_{G_2}(c)$ conflicts with $w_{L_{21}}(c)$ and $w_{L_{21}}(b)$ conflicts with $r_{G_1}(b)$). □

To ensure that the properties of global subschedules are preserved when global subtransactions are interleaved with local transactions, we must determine those properties of global subschedules which satisfy the following two conditions simultaneously:

(1) the correct execution of global transactions are maintained at the global level; and
(2) the properties must hold even if the execution of global transactions is interleaved with the operations of local transactions.

That is, a qualifying property of global subschedules must not only maintain the correct execution of global transactions but also tolerate the interaction of the execution of local transactions at local sites. Thus, condition (2) distinguishes global transaction management from other varieties of transaction management. Condition (2) may be defined more formally as follows:

Definition 3.1 (Local extensibility) *Let S be a global schedule in a given MDBS model. A property P of global subschedule S_G is* locally extensible *if, whenever P holds for S_G, P also holds when any $S'_G \leq S_G$ ($S'_G \leq S_G$ denotes "S'_G is a prefix of S_G") is interpolated by the operations of local transactions that follows the correctness criteria for execution of transactions at local sites.*

Thus, local extensibility defines precisely those conditions of global subschedules that must be satisfied to permit the unification of global and local transaction management without violation of local autonomy. The unifying principle is formulated as follows:

Definition 3.2 (Unifying principle) *The properties of global subschedules which maintain the correct execution of global transactions should be locally extensible.*

In the remainder of this paper, we will investigate the enforcement of local extensibility on global subschedules and the effects of such locally extensible properties of global subschedules on global schedules.

4 Maintaining Global Serializability

First, let us consider the hierarchical structure of global and local concurrency control in multidatabase systems. As we have seen, due to the constraints of local autonomy, local indirect conflicts may cause the execution order of global subtransactions to differ from their serialization order. Consequently, serialization may not be ensured on global schedules even if both global subschedule and local schedules are serializable. In [ZE93], a sufficient condition is proposed for the GTM to determine the serialization orders of global subtransactions at local sites. We summarize the main concepts as follows:

Definition 4.1 (Chain-conflicts) *A set $\mathcal{G}_k = \{G_{1k}, ..., G_{mk}\}$ of global subtransactions at local site LS_k is chain-conflicting if there is a total order $G_{i_1k}, G_{i_2k}, \cdots, G_{i_mk}$ on $\mathcal{G}_k$ such that $G_{i_1k} \overset{c}{\sim} G_{i_2k} \overset{c}{\sim} \cdots \overset{c}{\sim} G_{i_mk}$.*

The conflicting operations of $\mathcal{G}_k$ refer to those operations that determine the chain-conflicting relationships of global subtransactions in $\mathcal{G}_k$. If a set of global subtransactions at a local site is *chain-conflicting*, then the execution order of conflicting operations determines the serialization order of the global subtransactions.

Definition 4.2 (Chain-conflicting serializability) *A global subschedule S_G is* chain-conflicting serializable *if there is a total order O on G such that, for all local sites LS_k ($1 \leq k \leq m$), $\mathcal{G}_k$ is chain-conflicting in an order that is consistent with O and S_G is serializable in O.*

It is proven there that if the global subschedule is chain-conflicting serializable and the local schedules are serializable, the serialization order of global transactions can be synchronized at all local sites. Consequently, serializability can be ensured on global schedules. The central concern is that the execution order of conflicting operations of the global subtransactions can determine their serialization order. We now formally show that the serializability of chain-conflicting global subschedules is locally extensible.

Theorem 4.1 *The execution order of conflicting operations of global subtransactions ensures that their serialization order is persistent to the interactions of local transactions.*

Proof: Given a global schedule S and its global subschedule $S_{\mathcal{G}}$ with any G_1 and G_2 in $\mathcal{G}$, we need to show that if, for o_1 and o_2 are conflicting operations of G_{1k} and G_{2k} respectly and any $S'_{\mathcal{G}} \leq S_{\mathcal{G}}$, $o_1 \prec_{S'_{\mathcal{G}}} o_2$, then $G_{1k} \prec_{sr}^{S_k} G_{2k}$. Suppose $G_{1k} \not\prec_{sr}^{S_k} G_{2k}$. Then, since S_k is serializable, we must have $G_{2k} \prec_{sr}^{S_k} G_{1k}$. Since o_1 conflicts o_2, in any serial schedule S'_k which is conflict equivalent to S_k, $G_{2k} \prec_{S'_k} G_{1k}$, which implies $o_2 \prec_{S'_k} o_1$. Hence, $o_2 \prec_{S_k} o_1$. Consequently, $o_1 \not\prec_{S_k} o_2$. □

As a result, controlling the execution order of chain-conflicting operations of global subtransactions implies that the serialization orders of global subtransactions at local sites are determined at the global level. Since chain-conflicting serializability ensures that the execution order of chain-conflicting operations determines the serialization order of global subtransactions at each local site, the interactions of local transactions with global subschedules at local sites will not change the serialization order of global subtransactions. Thus, the serialization order of global subtransactions enforced by chain-conflicting serializability in a global subschedule is locally extensible.

Note that not all global transansactions can be arranged to be chain-conflicting. Neverthless, there are mechnisms that can enforce extra conflicts among global transactions, such as ticket method [GRS91].

The significance of maintaining serializability on global schedules is that if local and global transactions maintain both local and global integrity constraints, then a serializable global schedule is definitely correct. In general, however, due to local autonomy, local transactions may be totally unaware of global integrity constraints. In such situations, even a serializable global schedule would be incapable of maintaining correctness. Special treatment must be considered [GM91, RSK91]. We will not address such cases here. In the next subsection, we will investigate special situations in which global schedules are guaranteed to be correct.

5 Relaxing Global 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.

5.1 View-Based Two-Level Serializability

In order to avoid the potential of poor performance which may be caused by global serializability, several researches [DE89, MRKS91a] have suggested notions of correctness based on integrity constraints that are weaker than global serializability.

In [DE89], a 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.[2]

In [MRKS91a], another non-serializable criterion, termed *two-level serializability*, is proposed:

Definition 5.1 *(Two-level serializable global schedule) A global schedule is* two-level serializable, *denoted 2LSR, if its global subschedule and local schedules are serializable.*

As stated in [MRKS91b], the set of 2LSR global schedules is a superset of the set of QSR global schedules. Both criteria may not permit local/global integrity constraints that are defined among different sites. In addition, value dependencies are not allowed to be defined in global transactions. The following example given in [MRKS91b] is illustrative:

Example 5.1 *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 : $a := 1$

if $c > 0$ *then* $b := 1$

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

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

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

[2] See [BHG87] for the concept of conflict equivalence.

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.* □

We shall focus on two-level serializable global schedules and investigate conditions other than value dependencies which must be placed on global transactions to ensure that two-level serializable global schedules will preserve multidatabase consistency.

We propose a view-based two-level serializability criterion. The 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 [ZPB93].

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.

5.2 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 view

closed on a given set of data items.

Definition 5.2 *(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 and only 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 5.3 *(View closure of 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 $\mathcal{D}$ and t be a transaction in a schedule s. 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 in a schedule s is consistent if $read(t)$ is consistent. In the MDBS environment, the consistency of various views of transactions is defined as follows:

Definition 5.4 *(Local view consistency) A transaction t_i is* local view consistent *at LS_j if $read(t_i^{LD_j})$ is consistent.*

Definition 5.5 *(Global view consistency) A transaction t_i is* global view consistent *if $read(t_i^{GD})$ is consistent.*

5.3 Basic Lemmas

We shall now introduce the basic lemmas involved in the development of our theory.[3]

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

Lemma 5.1 *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: The proof of this lemma has been given in [RMB+93]. □

[3]To facilitate comparison, we shall largely adopt the system of notation used in [RMB+93].

As pointed out in [RMB+93], if lemma 5.1 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 [MRKS91b]:

Lemma 5.2 *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: 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 5.2 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 5.3 *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 $\{DS_1\}S\{DS_2\}$. For any $k = 1, ..., m$, if*

- *$(S^\tau)^{D_k}$ 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: 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 5.1, we see that $state(t_l, D_k, S, DS_1) \cup read(t_l)$ is consistent. By Lemma 5.2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. □

From Lemma 5.3, 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.

5.4 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 5.3 to a specific G_{rw} model and show that, when global transactions are local view consistent, global transactions read consistent data:

Lemma 5.4 *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: 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 $(S^G)^{GD}$. Since the global transactions are local view consistent, $read(t_i^{D-GD})$ is consistent for all $i = 1, ..., n$. By Lemma 5.3, $state(t_i, GD, S, DS_1)$ 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 5.1, $read(t_i)$ is consistent for all $i = 1, ..., n$. □

Following Lemma 5.4, 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 5.3 and 5.4 that, given that global transactions are local view consistent, local transactions read consistent data:

Corollary 5.1 *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: 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 5.4, $read(t_i^{D-LD_k})$ is consistent for all $i = 1, ..., n$. By Lemma 5.3, $state(t_i, LD_k, S, DS_1)$ 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 5.1 *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: 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 5.4 and Corollary 5.1, 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 5.3, $state(t_n, LD_k, S, DS)$ 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 5.2, $DS_4^{LD_k}$ 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 5.1, DS_2 is consistent. Hence, S is correct. □

In Example 5.1, 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 5.1 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.

5.5 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 [MRKS91a]:

Example 5.2 *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 : $b := 1$

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

p_2 : $a := 1$

$c := 1$

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 5.2, 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 5.4 still holds in this context; however, Corollary 5.1 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 5.5 *Let S be a 2LSR schedule in the $G_{rw}L_r$ 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: Since only the set $\mathcal{G}$ of global transactions write on global data items, we have $\tau_w(GD, S) \subseteq \mathcal{G}$. Since $S^{\mathcal{G}}$ is serializable, $(S^{\mathcal{G}})^{GD}$ is serializable. By Lemmas 5.3 and 5.4, $state(t_i, GD, S, DS_1)$ 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 5.6 *Let S be a 2LSR schedule in the $G_{rw}L_r$ 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: 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 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 5.2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. Now we consider t_l to be a global subtransaction. Following Lemma 5.5, $state(t_l, D_k, S, DS_1) \cup read(t_l)$ is consistent. Again, by Lemma 5.2, $state(t_{l+1}, D_k, S, DS_1)$ is consistent. □

The following theorem based upon Lemmas 5.5 and 5.6 illustrates the conditions under which 2LSR schedules preserve database consistency in the $G_{rw}L_r$ model.

Theorem 5.2 *Let S be a 2LSR schedule in the $G_{rw}L_r$ model with no integrity constraints present between local and global data items. If all global transactions in S are global view closed, then local view consistency implies that S is correct.*

Proof: 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 5.4, all global transactions read consistent data. Following Lemma

5.6, 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 5.6, $state(t_n, D_k, S, DS)$ 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 5.2, $DS_4^{LD_k}$ 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 5.1, DS_2 is consistent. Hence, S is correct. □

In Example 5.2, 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 5.2 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.

We now give the formal definition of view-based two-level serializability as follows:

Definition 5.6 *(View-based two-level serializability) A global schedule S is view-based two-level serializable if S is two-level serializable and all global transactions in S are local view consistent and view closed with respect to global data items.*

5.6 Maintaining View-Based Two-Level Serializability

In this section, we will discuss approaches to maintaining view-based two-level serializability. In order to ensure that a global schedule S is two-level serializable, the GTM needs to ensure only that S^G is serializable, since the LDBS at each local site LS_i ensures the serializability of S_i. Thus, in an MDBS, two-level serializability can easily be ensured, since global transactions are executed under the control of the GTM. For example, as pointed out in [MRKS91a], the GTM could follow a protocol very similar to 2PL [BHG87], in which the GTM maintains locks for every data item accessed by global transactions. It therefore remains only to design protocols for the enforcement of view closure and the detection of local view consistency for global transactions.

5.6.1 Enforcement of View Closure

Clearly, given any set of data items D, the computation of closure $cl(D)$ is straightforward. We now discuss the enforcement of view closure for global transactions.

After computing the closure of the global data items read by a global transaction, view closure can be easily enforced. This can be implemented by appending to the beginning of the global

transaction read operations on data items which are included in the closure but not read by the global transaction to the beginning of the global transaction.

5.6.2 Detection of Local View Inconsistency

The detection of inconsistency is a classical problem to which much attention has been directed [BB82, GW93]. When an update u is executed, it may cause a change of database state ST to ST_u. By applying tests derived from the constraints, the enforcement algorithm verifies that all relevant constraints hold in state ST_u.

Theorems 5.1 and 5.2 presented above have simplified consistency detection to the point that only local view inconsistencies of global transactions need be detected. As the read set of a global transaction usually involves only a few data items, the number of integrity constraints that must be checked is very limited. Thus, the detection of local view inconsistency becomes straightforward.

As stated in Theorems 5.1 and 5.2, local view consistency of global transactions must be ensured to maintain the correctness of two-level serializable global schedules. However, it is difficult to determine the set of integrity constraints that is satisfied by the read set of a global transaction when the global transaction is not view closed with respect to local data items. For example, assume a local integrity constraint $x > y$ and that global transaction G_i reads only x. G_i may be local view consistent if $x > y$ is ignored, but x may actually be part of an inconsistent local state. Following the proposed theory, such partially covered integrity constraints can be ignored.

6 Comparison of Different Criteria

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 [MRKS91a]. 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 [MRKS91a]. 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 [MRKS91a] 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.

[RMB+93] 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.

7 Conclusions

This paper has proposed new correctness criteria on the execution of local and global transactions, while value dependencies are permitted among the global subtransactions of a global transaction and compensation is used to preserve the semantic atomicity of global transactions.

Note that, in general, not every global subtransaction will be compensatable. In such a situation, other approaches may be combined with compensation. The proposed research on this aspect can be found in [BST90, MRKS92], where retry and redo approaches are used. While these approaches can be combined together to enhance global transaction management, it is not hard to see that in the combined approach, the criteria presented in this paper are still applicable.

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 theformulation 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.

In this paper, we have investigated the restrictions on global transaction management necessary to maintain the isolation of global transactions in the multidatabase environment without placing

restrictions on local sites. A unifying principle has been formulated to accommodate the need for autonomy on the part of integrated components in a multidatabase environment. The lack of global-level knowledge regarding these autonomous components requires that we consider the placing of restrictions on global transaction management. While most research work on multidatabase concurrency control has implicitly enforced this principle, the exact principle and its effects on multidatabase concurrency control has never before been formally defined and discussed.

References

[BB82] P. Bernstein and B. Blaustein. Fast Method for Testing Quantified Relational Calculus Assertions. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 39–50, Orlando, FL, 1982.

[BHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Databases Systems*. Addison-Wesley Publishing Co., 1987.

[BS88] Y. Breitbart and A. Silberschatz. Multidatabase Update Issues. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 135–142, June 1988.

[BST90] Y. Breitbart, A. Silberschatz, and G. Thompson. Reliable Transaction Management in a Multidatabase System. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 215–224, May 1990.

[BST92] Yuri Breitbart, Avi Silberschatz, and Glenn R. Thompson. Transaction managemnet issues in a failure-prone multidatabase system environment. *VLDB Journal*, 1(1):1–39, July 1992.

[DE89] 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, August 1989.

[DEK90] W. Du, A. Elmagarmid, and W. Kim. Effects of Local Autonomy on Heterogeneous Distributed Database Systems. Technical Report ACT-OODS-EI-059-90, MCC, February 1990.

[GM91] H. Garcia-Molina. Global consistency constraints considered harmful for heterogeneous database systems. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pages 248–250, Kobe, Japan, April 1991.

[GMK88] H. Garcia-Molina and B. Kogan. Node Autonomy in Distributed Systems. In *Proceedings of the First International Symposium on Databases for Parallel and Distributed Systems*, pages 158–166, Austin, Texas, December 1988.

[Gra81] J. Gray. The transaction concept: Virtues and limitations. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–154, Cannes, France, September 1981.

[GRS91] 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, April 1991.

[GW93] A. Gupta and J. Widom. Local Verification of Global Integrity Constraints in Distributed Databases. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 49–58, Washington, DC, May 1993.

[Had88] V. Hadzilacos. A theory of reliability in database systems. *Journal of the Association for Computing Machinery*, 35(1):121–145, January 1988.

[HR83] T. Haerder and A. Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4), July 1983.

[Lit86] W. Litwin. A multidatabase interoperability. *IEEE Computer*, 19(12):10–18, December 1986.

[MRB+92] 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.

[MRKS91a] 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.

[MRKS91b] 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, December 1991.

[MRKS92] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. A transaction model for multidatabase systems. In *Proceedings of International Conference on Distributed Computing Systems*, June 1992.

[ÖV91] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, Inc., 1991.

[Pu88] C. Pu. Superdatabases for Composition of Heterogeneous Databases. In *Proceedings of the International Conference on Data Engineering*, pages 548–555, February 1988.

[RMB+93] 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.

[RSK91] M. Rusinkiewicz, Amit Sheth, and G. Karabatis. Specifying interdatabase dependencies in a multidatabase environment. *IEEE Computer*, 24(12), December 1991.

[Vei90] J. Veijalainen. *Transaction Concepts in Autonomous Database Environments.* R. Oldenbourg Verlag, Germany, 1990.

[VW92] J. Veijalainen and A. Wolski. Prepare and commit certification for decentralized transaction management in rigorous heterogeneous multidatabases. In *Proceedings of the International Conference On Data Engineering*, 1992.

[ZE93] Aidong Zhang and Ahmed Elmagarmid. A theory of global concurrency control in multidatabase systems. *The VLDB Journal*, 2(3):331–359, July 1993.

[ZPB93] Aidong Zhang, Evaggelia Pitoura, and Bharat Bhargava. A View-Based Approach to Relaxing Serializability in Multidatabase Systems. Technical Report CSD-TR-93-082, Purdue University, December 1993.