1993

A Unifying Approach to Hierarchical Transaction Management in Multidatabase Systems

Ahmed K. Almagarmid

Aidong Zhang

Report Number:
93-012

A UNIFYING APPROACH TO HIERARCHICAL TRANSACTION MANAGEMENT IN MULTIDATABASE SYSTEMS

Ahmed K. Elmagarmid
Aidong Zhang

CSD-TR-93-012
January 1993
(Revised 3/93)

A Unifying Approach to Hierarchical Transaction Management in Multidatabase Systems *

Ahmed K. Elmagarmid and Aidong Zhang
Department of Computer Science
Purdue University
West Lafayette, IN 47907 USA
{ake, azhang}@cs.purdue.edu

Technical Report No: CSD-TR-93-012

Abstract

A multidatabase transaction management system is a two-level hierarchical approach to the global integration of local autonomous transaction management systems. This paper offers an approach to the correct execution of both global and local transactions without violation of local autonomy. Local extensibility is proposed as a unifying principle in the combination of global and local transaction management. Global consistency is maintained through a global concurrency control correctness criterion which is less restrictive than global serializability. 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.

1 Introduction

This paper investigates those problems that arise in multidatabase systems at the level of transaction management. The role of a transaction manager is the preservation of the atomicity, isolation, and durability [ÖV91, AA92] of transactions. In a multidatabase model, transaction management

*Zhang is supported by a Purdue Research Foundation Fellowship and Elmagarmid is supported by the NSF under grant IRI-8857952.

is handled at both the global and local levels. A global transaction manager is superimposed upon a set of local autonomous database systems. 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[1]. The difficulty of maintaining global serializability in multidatabase systems has been made evident in the recent literature [AGMS87, BS88, Pu88, DE89, GRS91, VW92]. To preserve the isolation of global transactions without violation of local autonomy, a global concurrency controller or scheduler must be a component of the global transaction management system. This controller ensures the correct execution of global transactions while allowing such executions to interleave with the globally uncontrolled execution of local transactions at local sites. 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.

Preserving the atomicity of global transactions in multidatabase systems has been recognized as an open and difficult issue [SSU91]. The goal of atomic commitment is to ensure that either all or none of the effects of each transaction are made permanent. The traditional two-phase commitment

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

(2PC) developed in distributed database environments has been shown [LKS91, SKS91, MRKS92] to be inadequate to the preservation of the atomicity of global transactions in the multidatabase environment. For example, some local database systems may not support a visible prepare-to-commit state. Even when the local database systems do provide such support, the potential blocking and long delays of 2PC would severely degrade local execution autonomy.

In this paper, we shall study the principles of global concurrency control and atomic commitment 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 global concurrency control and atomic commitment in the multidatabase environment without placing additional restrictions on local database systems. This 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 and its role in the maintenance of global consistency. 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.

The body of this paper is organized as follows. Section 2 introduces the system model and defines the terminology to be employed. Section 3 discusses the central concerns of this research and surveys related work. In Section 4, a unifying principle for global transaction management is proposed and its enforcement is analyzed. Section 5 advances the concept of MDBS-serializability, a less restrictive correctness criterion than global serializability for the maintenance of global consistency. A discussion of the related issues and concluding remarks are given in Section 6 and 7.

2 The System 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

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. Global transactions are submitted to the GTM, while local transactions are submitted to the LDBSs. Figure 1 illustrates this model.

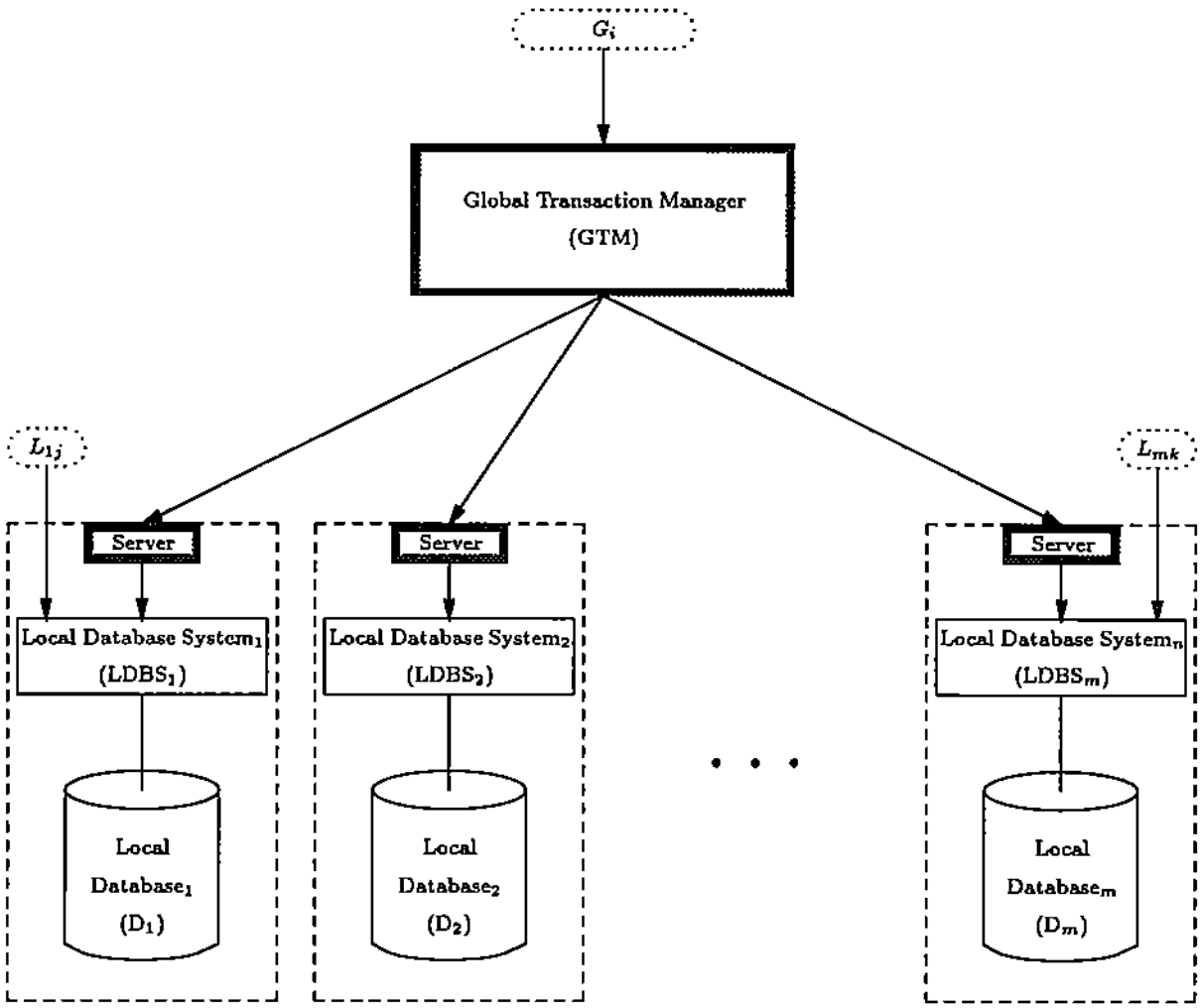

Figure 1: Two-level conceptual multidatabase architecture.

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 control the execution order of global

transactions (defined in the next subsection) by controlling their submission order.

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

2.2 Notation and Terminology

For the elements of a transaction, we assume the availability of four basic 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]. 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 where each global subtransaction is a transaction accessing the data items at a single local site. A global transaction $G_{ij}^{(k)}$ denotes a global subtransaction of G_i accessing LDBS$_j$. A global transaction may have more than one global subtransaction at a single local site. A set $\mathcal{G} = \{G_1, \cdots, G_n\}$ contains those global transactions that are submitted to the GTM, and $\mathcal{G}_k$ denotes the set of global subtransactions of $\mathcal{G}$ at local site LS_k. A transaction T refers to either a local or global transaction, while OP_T denotes the set of operations contained in 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 $o_j \in OP_{T_j}$.

Without loss of generality, let global transaction $G_i = \{G_{i1}, G_{i2}, \cdots, G_{im}\}$, where G_{ij} is the global subtransaction at local site LS_j. We say that G_{ij_t} is *value-dependent* on $G_{ij_1}, ..., G_{ij_{t-1}}$ $(1 \leq j_1, ..., j_t \leq m)$ if the execution of one or more operations in G_{ij_t} is semantically determined by the values read by $G_{ij_1}, ..., G_{ij_{t-1}}$.

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 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_G is S restricted to the set G of global transactions in S. A lower case s refers to either a local or global schedule.

We say that a schedule s is *serial* if the operations of different transactions in s are not interleaved. We say that the execution of T_1 precedes the execution of T_2 in schedule s if all operations of T_1 are executed before any operation of T_2 in s. Obviously, a total execution order can be determined on transactions in a serial schedule. A global schedule S is globally serializable if and only if S is serializable [BHG87, Had88] in a total order on both committed global and local transactions in S. We denote $o_1 \prec^s_{eo} o_2$ 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.

Beyond the conventional criterion of serializability, we must also define the notion of consistency as it is applied in this paper. Following the traditional approach, a database state is defined as a mapping of every data item to a value of its domain, and the integrity constraints on these data items are used to define database consistency. A database state is considered to be *consistent* and a schedule is *correct* if it preserves these database integrity constraints. A schedule is *strongly correct* [MRKS91] if it is correct and each transaction in the schedule sees only consistent states. In a multidatabase system, there are two types of integrity constraints; local integrity constraints are defined on data items in a single local site, while global integrity constraints are defined on data items in different local sites. Local schedules must preserve local integrity constraints, while global schedules must preserve both local and global integrity constraints. We assume that each local (global) transaction is consistent; that is, its execution trransforms a local (global) database from one consistent state to another.

3 Central Issues and Related Research

In this section, we shall discuss the central obstacles to effective multidatabase transaction management and survey the research in this area.

3.1 Local Indirect Conflicts

The hierarchical architecture of our model predicates a hierarchical concurrency control structure, which in turn renders difficult the maintenance of global serializability. It has been shown in [MRB+92] that, if each global transaction has only one global subtransaction at each local site

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[2], then S is globally serializable. Therefore, the maintenance of global serializability can be reduced to the synchronization at all local sites of the relative serialization orders of the global subtransactions of each global transaction. That is, for any two global transactions G_i and G_j, the serialization orders of all global subtransactions of G_i either precede or follow the serialization orders of all global subtransactions of G_j at local sites. Since global subtransactions are received by the local concurrency controller and treated as local transactions, the global concurrency controller must reflect the resulting serialization orders in a manner which is consistent with its local counterparts. We must therefore seek an approach by which the global concurrency controller can determine the serialization orders of global subtransactions at each local site without violation of local autonomy.

In general, the global concurrency controller possesses no information regarding local serialization orders, and the execution orders of global subtransactions may in fact differ from their serialization orders at local sites. It has been pointed out [DE89, GRS91] that local indirect conflict is the major factor in these discrepancies. Example 1 illustrates this situation.

Example 1 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1 and b,c are in D_2. The following global transactions are submitted:*

$G_1 : w_{G_1}(a) r_{G_1}(b), \quad G_2 : r_{G_2}(a) w_{G_2}(c)$

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:

$S_1 : w_{G_1}(a) r_{G_2}(a)$

$S_2 : w_{L_{21}}(b) r_{G_1}(b) w_{G_2}(c) w_{L_{21}}(c)$

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 $r_{G_1}(b)$). □

[2]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 same order symbol as that of the global transaction to which it belongs.

Thus, even though the execution orders of the global subtransactions at all local sites are serial and consistent, they may differ from their serialization orders in local schedules because of local indirect conflicts. Consequently, global serializability is not maintained. Local indirect conflict is thus the major obstacle to achieving global serializability in MDBSs. Unfortunately, it is impossible to predict local indirect conflicts at the global level without violation of local autonomy, since the GTM has no knowledge of the submissions of local transactions.

Some approaches to the above issue propose the relaxation of global serializability theory as a means to simplify global concurrency control. These approaches, such as quasi-serializability [DE89] and two-level serializability [MRKS91], can maintain global consistency in restricted applications. For example, quasi-serializability requires that no value-dependency between global subtransactions be allowed, and restricted Read-Write models are employed in two-level serializability. Other methods utilize local serialization information contained in local concurrency control protocols. These approaches, such as rigorous local schedules [BGRS91], strongly recoverable local schedules [BS92, RAZ92], or serialization events at local sites [ED90, MRB+92, Pu88] have also achieved some initial success. If the pre-existing local transaction management systems satisfy these conditions, then these methods are applicable. The Optimistic Ticket Method (OTM) proposed in [GRS91] is the first to successfully show that the serialization order of global subtransactions in a local site can be determined at the global level without violation of local autonomy.

3.2 Local Unilateral Aborts

A two-phase commit protocol has been proposed for traditional distributed database systems to ensure the atomicity of transactions. This protocol relies on the ability of local database systems to support a prepare-to-commit state, in which a transaction has not yet been committed but is guaranteed the ability to commit. However, most database systems in current use do not support a visible prepare-to-commit state. In these instances, a local database system that participates in a multidatabase environment may unilaterally abort a global subtransaction without agreement from the global level. It may be a violation of local autonomy to require such local database systems to provide prepare-to-commit states.

To address this obstacle, approaches utilizing the redo, retry, and compensation techniques have been proposed. The redo approach proposed in [BST90] acts as a pseudo-2PC, with servers rather than the local LDBSs considered as the participants. If a global subtransaction is aborted by a

LDBS after the GTM has decided to commit the global transaction, the server at the abort site submits a *redo transaction* to the LDBS for execution. This redo transaction consists of all the write operations performed by the global subtransaction. Inconsistencies may arise if some local transactions are executed after a global subtransaction is aborted and before its redo operations are executed. Thus, the redo approach requires that the data items accessed by local transactions must be different from the data items accessed by global subtransactions at a local site.

The retry approach requires the resubmission of an aborted global subtransaction to the LDBS. It is assumed that the global subtransaction will commit after it is retried a sufficient number of times. Some difficulties may arise, however, if one global subtransaction has a value-dependency relationship with another global subtransaction. If, due to local autonomy, the global transaction manager cannot block local transactions from execution after a global subtransaction aborts but before it is retried, then execution of such local transactions may result either in the inability to retry the aborted global subtransaction or in the generation of database inconsistency if it is retried [MRKS92]. To overcome this inconsistency, an approach is proposed in [MRKS92] that stipulates that no value-dependencies may exist between the subtransactions of a global transaction.

The concept of compensation, which was originally proposed [GM83, GMS87] to address the semantic atomicity of long-running transactions, has been shown [LKS91] to be useful in the multidatabase system environment. In this approach, the global subtransactions of a global transaction are allowed to commit unilaterally at local sites. Semantic atomicity guarantees that, if all global subtransactions commit, then the global transaction commits; otherwise, all tentatively committed global subtransactions are compensated. The compensation approach requires that the local transactions to be executed at a local site must be commutative with the executing global subtransaction before its commitment, or its compensating transaction may be executed.

In summary, the central obstacle to global transaction management is the inability of the global level to predict the interactive ramifications of local transaction management and their potential impact on global subschedules. In this paper, we propose the placement of restrictions on the global level to circumvent such local interactions.

4 The Unifying Principle

In this section, we shall advance a principle that unifies global and local transaction management so as to maintain the isolation and atomicity of global transactions in a multidatabase environment

without violation of local autonomy. We then discuss the application of this theory to global transaction management.

4.1 Local Extensibility

As we have seen from the discussion of the previous section, 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 environment. We may term this approach the unifying principle.

Let S be a global schedule and the global subschedule $S_{\mathcal{G}}$ be S restricted on the set $\mathcal{G}$ of global transactions in S. 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 isolation and atomicity 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 correctness of the execution of local transactions is already guaranteed at local sites. Thus, the main focus of multidatabase transaction management must be to maintain the correctness of execution of global transactions in respect to the globally uncontrolled local interactions at local sites. 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:

- **Condition 1**: the isolation and atomicity of global transactions are preserved at the global level; and
- **Condition 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 isolation and atomicity 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 1 (Local extensibility) *Let S be a global schedule. A property P of global subschedule S_G is* locally extensible *if and only if, whenever P holds for S_G, P also holds for S_G in S.*

Definition 1 states that a locally extensible property of global subschedules allows the GTM to have an effect on global schedules. In order to effect a compromise among locally autonomous components without imposing restrictions, the GTM must find some such means to affect, and thus operate through, global schedules. Otherwise, the interaction of local transactions may in some manner change the properties of global subschedules, presenting obstacles to the investment of global schedules with novel properties. For instance, in Example 1, S_G is serializable. However, the serializability property is not locally extensible. The interactions of $w_{L_{21}}(b)$ and $w_{L_{21}}(c)$ with S_G cause S_G to be non-serializable in S. If G_2 is instead defined as $r_{G_2}(a)w_{G_2}(c)w_{G_2}(b)$, then at local site LS_2, after $w_{L_{21}}(b)r_{G_1}(b)$ is scheduled, $w_{L_{21}}(c)$ must be scheduled before $w_{G_2}(c)$ to maintain local serializability. Hence, the correct schedule for S_2 is:

$S_2 : w_{L_{21}}(b)r_{G_1}(b)w_{L_{21}}(c)w_{G_2}(c)w_{G_2}(b)$,

which implies $G_1 \prec_{sr}^{S_2} G_2$. The serializability property of S_G is now locally extensible, regardless of any interleaving of the operations of local transactions.

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:

> **Unifying principle:** The properties of global subschedules which maintain the isolation and atomicity of global transactions must be locally extensible.

We shall investigate below the enforcement of local extensibility on global subschedules and the effects of such locally extensible properties of global subschedules on global schedules.

4.2 Enforcement of Local Extensibility

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. Thus, the GTM may not be able to generate all global schedules which satisfies the sufficient condition stated in Section 3.1 (see also [MRB+92]). In [ZE93a, ZE93b], a sufficient condition

is proposed for the GTM to determine the serialization order of global subtransactions at local sites. 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. 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 $\mathcal{G}_k$. The following example is illustrative:

Example 2 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1 and b, c are in D_2. The following global transactions are submitted:*

$$G_1 : r_{G_1}(a) r_{G_1}(b), \quad G_2 : w_{G_2}(c) w_{G_2}(b), \quad G_3 : w_{G_3}(a) r_{G_3}(c),$$

where at local site LS_1, $G_{11} \overset{c}{\sim} G_{31}$, and at local site LS_2, $G_{12} \overset{c}{\sim} G_{22} \overset{c}{\sim} G_{32}$. If the execution orders of chain-conflicting operations which determine the chain-conflicting relationships of global subtransactions are maintained as:

$$r_{G_1}(a) \prec_{co}^{S_1} w_{G_3}(a)$$

$$r_{G_1}(b) \prec_{co}^{S_2} w_{G_2}(b)$$

$$w_{G_2}(c) \prec_{co}^{S_2} r_{G_3}(c)$$

then the serialization order of global subtransactions at local site LS_1 is always $G_{11} \rightarrow G_{31}$ and the serialization order of global subtransactions at local site LS_2 is always $G_{12} \rightarrow G_{22} \rightarrow G_{32}$, no matter what local transactions are executed at local sites. □

Thus, 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. It has also been shown in [ZE93b] that chain-conflicting relationships of global subtransactions define the weakest conflicting condition for the GTM to determine the serialization order of global subtransactions at local sites without violation of local autonomy.

We say a global subschedule $S_\mathcal{G}$ is *chain-conflicting serializable* if $\mathcal{G}$ is chain-conflicting in an order O and $S_\mathcal{G}$ is serializable in O.

From the above, if each global transaction has only one global subtransaction at each local site, then the enforcement of *chain-conflicting serializability* on global subschedules implies that all serialization orders of global subtransactions at all local sites are consistent with the same order. Thus, global serializability is maintained.

Since chain-conflicting serializability ensures that the execution order of chain-conflicting operations determines the serialization order of global subtransactions at each local site, local indirect conflicts are no longer a concern. 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.

In order to enforce a chain-conflicting relationship on global transactions, an *extra operation method* is suggested to effect chain-conflicting relationships among global subtransactions. Let G_{ik} and G_{jk} be nonconflicting global subtransactions at local site LS_k. Conflicts among global transactions can then be simulated. Suppose G_{ik} is executed before G_{jk}. If G_{ik} and G_{jk} do not conflict and an operation of G_{ik} is on data item x, we then append operations $r(x)$ and $w(x)$ to G_{jk}. Let G'_{jk} denote G_{jk} after appending these extra operations. G_{ik} and G'_{jk} now conflict with each other, and the effect on D_k made by G'_{jk} remains the same as that made by G_{jk}. One advantage of the extra operation method is that it requires nothing from local sites. In addition, the conflict relationships generated by the extra operation method are weaker than those generated by the ticket method [GRS91]; $G_1 \overset{c}{\sim} G_2 \overset{c}{\sim} G_3$ may not imply $G_1 \overset{c}{\sim} G_3$.

We shall now discuss the enforcement of local extensibility on the atomic commitment of global transactions. As discussed earlier, the main obstacle to the maintenance of atomic commitment is the handling of local unilateral aborts. We explore here the condition on the commitment order of global subtransactions such that each aborted global subtransaction will be retriable. We define that an aborted global subtransaction is *retriable* if it can be resubmitted for execution without creating any inconsistencies in the involved local databases, regardless of whatever operations have been executed at local sites. There are two obstacles that may restrict the retriability of global subtransactions. The first arises when global subtransactions are related by value-dependencies. For instance, let us assume that a value written by G_{il} at local site LS_l is value-dependent on a value read by G_{ij} at local site LS_j. If G_{il} commits and G_{ij} aborts, then the retrial of G_{ij} may result in inconsistencies between the data read from the original execution of G_{ij} and from its retrial, since local transactions may be executed after G_{ij} is aborted but before it is retried at LS_j. The second obstacle arises when the serialization orders of global subtransactions at all local sites must be synchronized. If a global subtransaction is aborted and then resubmitted when a global subtransaction initially serialized after it has already committed, the serialization order of global subtransactions may as a result be different from their original serialization order at a given local

site. This order, in turn, may be inconsistent with the serialization order of global transactions that all local sites have agreed to enforce. To ensure the general retriability of global subtransactions, we propose the following sequential commitment rules:

- **Rule 1**: each global subtransaction must commit after all global subtransactions upon which it is value-dependent have committed; and
- **Rule 2**: at each local site, the commitment order of global subtransactions must be consistent with their serialization order.

Rule 1 ensures the retriability of each global subtransaction relative to other global subtransactions that belong to the same global transaction. If the execution of a retried global subtransaction leads to a result which is different from that of its original execution, then those global subtransactions which are value-dependent upon it may be aborted and re-executed. Consequently, a global subtransaction remains retriable as long as all other global subtransactions that are value-dependent upon it have not committed. Rule 2 ensures the retriability of each global subtransaction relative to global subtransactions which belong to different global transactions. Those global subtransactions which are serialized after the aborted global subtransaction can be aborted and re-executed in order to preserve the synchronized relative serialization order of global subtransactions at a local site[3]. This second rule is necessary to effect a compromise between atomic commitment and concurrency control. Note that the GTM can control the commitment order of global subtransactions by controlling their submissions at the global level without placing any restrictions on local sites.

Rules 1 and 2 therefore ensure that the commitment order of global subtransactions is locally extensible. Local interactions will have no effect on the commitment order of global subtransactions. An atomic commitment approach which follows the sequential commitment rules and retrys the aborted global subtransactions is termed a *sequential commit-retry* approach. The sequential commit-retry approach ensures the atomicity of global transactions, provided that each global subtransaction commits after it is retried a sufficient number of times.

Following from Rule 1, the acyclicity of value-dependency relationships defined on global transactions provides the necessary and sufficient conditions for maintaining a set of global transactions as atomic, using the sequential commit-retry approach. To satisfy the requirement for acyclicity, any two component global subtransactions of a global transaction may not be mutually value-

[3]An efficient implementation of these rules which avoids cascading aborts appears in [ZCEB93].

dependent, either directly or indirectly (through other global subtransactions). The conventional assumption that each global transaction has only one global subtransaction at each local site may be too restrictive to permit the sequential commit-retry approach to be employed in many applications. For instance, assume that two global subtransactions G_{il} and G_{ik} of a global transaction G_i at local sites LS_l and LS_k have the following value-dependencies: (1) G_{il} reads a and G_{ik} uses this value to compute and write to c, and (2) G_{ik} reads d, and this value is used by G_{il} to compute and write to b. G_{il} and G_{ik} are value-dependent upon each other. We observe that the decomposition of global subtransactions may break such cycle of value-dependency. However, this method may cause each global transaction to have more than one global subtransaction at a local site, undermining the maintenance of global serializability. The following example is illustrative:

Example 3 *Let $G_i = \{G_{i1}, G_{i2}\}$ be a global transaction, with G_{i1} and G_{i2} being mutually value-dependent. If one-step decomposition on G_{i1} produces $G_{i1}^{(1)}$ and $G_{i1}^{(2)}$, such that G_{i2} is value-dependent on $G_{i1}^{(1)}$, $G_{i1}^{(2)}$ is value-dependent on G_{i2}, and no value-dependency is defined between $G_{i1}^{(1)}$ and $G_{i1}^{(2)}$, then the value-dependency relationships among the global subtransactions of G_i are acyclic. As a result, the commitment order of G_i in the global subschedule must be $G_{i1}^{(1)} \rightarrow G_{i2} \rightarrow G_{i1}^{(2)}$, which follows rule 1. However, the LDBS at LS_1 will consider $G_{i1}^{(1)}$ and $G_{i1}^{(2)}$ to be two different local transactions. Even though the GTM can ensure that $G_{i1}^{(1)}$ is serialized immediately before $G_{i1}^{(2)}$, there may be a local transaction L which is locally serialized between $G_{i1}^{(1)}$ and $G_{i1}^{(2)}$; that is, $G_{i1}^{(1)} \prec_{sr}^{S_1} L \prec_{sr}^{S_1} G_{i1}^{(2)}$. Consequently, global serializability is not maintained.* □

Successful application of the decomposition technique is therefore predicated upon the relaxation of global serializability in circumstances such as that illustrated above. This possibility is discussed in the next section.

5 Relaxation of Global Serializability

In this section, we address the issue of maintaining global consistency while allowing each global transaction to have more than one global subtransaction at each local site. We assume that the atomicity of global transactions is preserved. A correctness criterion, termed *MDBS-serializability*, is proposed. This criterion is less restrictive than global serializability and can guarantee global consistency.

Let $G_{ij} = \{G_{ij}^{(1)} ... G_{ij}^{(k)}\}$ denote that global transaction G_i has k global subtransactions at local

site LS_j, with the execution order of these global subtransactions determined by their semantics. Each global subtransaction must be *locally consistent*; that is, its execution transforms a local database from one local consistent state to another, since the LDBSs treat each global subtransaction as an independent local transaction. At the global level and with regard to the concurrency control of global transactions, G_i is still treated as a single global transaction, since global integrity constraints must still be preserved. We have seen that global serializability cannot be maintained if a global transaction is allowed more than one global subtransaction at a local site. We therefore advance an alternative correctness criterion which preserves global consistency.

Section 4 has shown that the serialization order of global subtransactions enforced by chain-conflicting serializability on the execution of global transactions is locally extensible. This principle can be applied here, so that the relative serialization orders of global subtransactions can be synchronized at all local sites using chain-conflicting serializability at the global level, even if each global transaction has more than one global subtransaction at each local site. Building upon this, we claim that if chain-conflicting serializability is ensured at the global level, then a correctness criterion on global schedules which is less restrictive than global serializability can be obtained. This observation is first illustrated as follows.

Let $G_1 = \{\{G_{11}^{(1)}G_{11}^{(2)}\}, G_{12}\}$ and $G_2 = \{G_{21}, G_{22}\}$, where $G_{11}^{(1)}, G_{11}^{(2)}, G_{21}$ are at local site LS_1 and $G_{12}^{(1)}, G_{12}^{(2)}, G_{22}$ are at local site LS_2. Without loss of generality, let G_1 precede G_2 in their chain-conflicting related order. Thus, chain-conflicting serializability determines that $G_{11}^{(1)} \prec_{sr}^{S_1} G_{11}^{(2)} \prec_{sr}^{S_1} G_{21}$ and $G_{12} \prec_{sr}^{S_2} G_{22}$. At local site LS_1, there may exist a local transaction L_1 such that $G_{11}^{(1)} \prec_{sr}^{S_1} L_1 \prec_{sr}^{S_1} G_{11}^{(2)} \prec_{sr}^{S_1} G_{21}$, which results in a non-serializable global schedule of L_1, G_1, and G_2. However, since $G_{11}^{(1)}$ and $G_{11}^{(2)}$ are locally consistent transactions at LS_1, the interleavings of $G_{11}^{(1)}$, L_1, and $G_{11}^{(2)}$ will not violate any integrity constraints. A bank application illustrates this situation:

Example 4 *Consider a banking database located at two local sites: local site LS_1, with accounts a and b, and local site LS_2, with account c. Global transaction G_1 transfers \$100 from account a to account c and reads the balance of a and b, while global transaction G_2 reads the balances of both a and c, and local transaction L_1 transfers \$100 from account a to account b:*

```
G1: if a > 100 then a := a-100 else abort;
    c := c+100;
    read(a,b).
```

```
G2: read(a,c).
L1: if a > 100 then a := a-100 else abort;
      b := b+100.
```

Consider the following global schedule S of G_1, G_2, and L_1:

$S_1 : r_{G_{11}^{(1)}}(a) w_{G_{11}^{(1)}}(a) r_{L_1}(a) w_{L_1}(a) r_{L_1}(b) w_{L_1}(b) r_{G_{11}^{(2)}}(a) r_{G_{11}^{(2)}}(b) r_{G_{21}}(a),$

$S_2 : r_{G_{12}}(c) w_{G_{12}}(c) r_{G_{22}}(c),$

where, at LS_1, $G_{11}^{(1)} \prec_{sr}^{S_1} L_1 \prec_{sr}^{S_1} G_{11}^{(2)} \prec_{sr}^{S_1} G_{21}$. Clearly, S is not globally serializable. However, S is correct, since it does not violate any integrity constraints to be defined. □

MDBS-serializability is defined as follows:

Definition 2 (MDBS-serializability) *A global schedule S is* MDBS-serializable *if S is serializable without considering the local transactions which are serialized between two global subtransactions that belong to a single global transaction at a local site.*

We now formally show that if each global subtransaction is locally consistent, then MDBS-serializable schedules are correct. We shall consider the most complicated situation, in which all data items at each local site are accessible by both local transactions and global subtransactions. A *composite* transaction is defined as a global transaction into which local transactions are inserted between its locally consistent global subtransactions at a local site. We first demonstrate that a composite transaction is a *legal* global transaction; that is, its execution transforms a multidatabase from one globally consistent state to another. In this scenario, we assume that local transactions do not violate any global integrity constraints[4]. Thus, certain global integrity constraints would not exist. For instance, in Example 4, if there is a global integrity constraint $a < c$ and no any local integrity constraints at the two local sites, then a local transaction updating a may violate the global integrity constraint. In such cases, even a globally serializable schedule cannot maintain global consistency.

Since a global subtransaction is both locally consistent and treated like a local transaction, its execution must result in the local database state being part of a globally consistent state. Otherwise, an equivalent local transaction will cause a globally inconsistent state, contradicting

[4]If there exist interdependencies [RSK91], then local transactions may violate global integrity constraints, and special methods must be applied [RS91]. Such issues will not be addressed here.

our initial assumption. For instance, in Example 4, a local transaction which withdraws \$100 from a is equivalent to a global subtransaction which performs the same task. However, the semantics of these two transactions might be different. The local transaction may spend the \$100 withdrawn, while the global subtransaction may be part of a transfer of funds from a to c. Though another global subtransaction which deposits \$100 into c must be executed to preserve global consistency, the execution of the global subtransaction involving withdrawal leaves the local database state as part of a globally consistent state. Thus, the interleaving of locally consistent global subtransactions and local transactions violates no local or global integrity constraints. The following observation is an immediate consequence of this situation:

Proposition 1 *A composite transaction is a legal global transaction.*

Proof: Without loss of generality, let global transaction G_1 be:

$$G_1 = G_{11}^{(1)}...G_{11}^{(n_{11})}...G_{1j}^{(1)}...G_{1j}^{(n_{1j})}...G_{1m}^{(1)}...G_{1m}^{(n_{1m})},$$

where n_{1j} $(1 \leq j \leq m)$ denotes the number of global subtransactions of G_1 at local site LS_j and each global subtransaction of G_1 is locally consistent. Let T_1 be a composite transaction of G_1:

$$T_1 = G_{11}^{(1)}L_{11}...G_{11}^{(n_{11})}...G_{1j}^{(1)}L_{1j}...G_{1j}^{(n_{1j})}...G_{1m}^{(1)}L_{1m}...G_{1m}^{(n_{1m})}.$$

We need to prove that the interleaving of global subtransactions and local transactions in T_1 at each local site will not violate any local or global integrity constraints. It is obvious that T_1 violates no local integrity constraints, since each global subtransaction or local transaction is locally consistent. We have also assumed that, in our scenario, local transactions violate no global integrity constraints. The outstanding question is now whether the execution of a locally consistent global subtransaction $G_{1i}^{(t)}$ $(1 \leq t \leq n_{1i})$ at LS_i results in a local database state which is globally inconsistent, assuming that all $G_{1j}^{(t)}$ $(1 \leq t \leq n_{1j}$, $1 \leq j \leq m$, and $j \neq i)$ at other local sites have been executed.

Clearly, the execution of all subtransactions $G_{1i}^{(1)}...G_{1i}^{(n_{1i})}$ of G_1 at LS_i results in a local database state which is part of a globally consistent state, and global consistency is preserved when the global subtransactions of G_1 at other local sites are also executed. Suppose that $G_{1i}^{(t)}$ results in a globally inconsistent state which must be remedied by other global subtransactions at the same local site. The effect of such a global subtransaction can also be simulated by a local transaction, which implies that a local transaction may violate global integrity constraints, contradicting our earlier assumption. Therefore, no global subtransaction in G_1 can itself result in a globally inconsistent

state when the effect of the global subtransactions of G_1 at other local sites have been considered. Hence, the interleaving of global subtransactions and local transactions in T_1 violate no local or global integrity constraints. T_1 is therefore a legal global transaction. □

As a consequence, all local transactions which are serialized between the locally consistent global subtransactions of a single global transaction can be treated as part of this global transaction in verifying the correctness of global schedules. If every local transaction which is serialized between two global subtransactions that belong to a single global transaction at a local site can be treated as part of a global transaction, then MDBS-serializable schedules are effectively equivalent to globally serial schedules. In Example 4, the MDBS-serializable schedule of L_1, G_1, and G_2 is effectively equivalent to the serial schedule $G_1'G_2$, where global transaction G_1' consists of $G_{11}^{(1)}L_1G_{11}^{(2)}G_{12}$. Consequently, the following theorem is obtained:

Theorem 1 *If all global subtransactions are locally consistent, then MDBS-serializable global schedules are correct.*

Proof: Let S be an MDBS-serializable global schedule and $\mathcal{G} = \{G_1, ..., G_k\}$. Without loss of generality, let

$$G_1 = G_{11}^{(1)}...G_{11}^{(n_{11})}...G_{1m}^{(1)}...G_{1m}^{(n_{1m})},$$

...

$$G_i = G_{i1}^{(1)}...G_{i1}^{(n_{i1})}...G_{ij}^{(1)}...G_{ij}^{(n_{ij})}...G_{im}^{(1)}...G_{im}^{(n_{im})},$$

...

$$G_k = G_{k1}^{(1)}...G_{k1}^{(n_{k1})}...G_{km}^{(1)}...G_{km}^{(n_{km})},$$

and S be conflict equivalent to S':

$$S' = L_{10}\underbrace{G_{11}^{(1)}L_{11}...G_{11}^{(n_{11})}...G_{1m}^{(1)}L_{1m}...G_{1m}^{(n_{1m})}}_{G_1'}...$$
$$L_{i0}\underbrace{G_{i1}^{(1)}L_{i1}...G_{i1}^{(n_{i1})}...G_{ij}^{(1)}L_{ij}...G_{ij}^{(n_{ij})}...G_{im}^{(1)}L_{im}...G_{im}^{(n_{im})}}_{G_i'}...$$
$$L_{k0}\underbrace{G_{k1}^{(1)}L_{k1}...G_{k1}^{(n_{k1})}...G_{km}^{(1)}L_{km}...G_{km}^{(n_{km})}}_{G_k'},$$

where n_{ij} $(1 \leq i \leq k, 1 \leq j \leq m)$ denotes the number of global subtransactions of G_i at local site LS_j and L refers to local transactions. Since all global subtransactions are locally consistent,

$G'_1, ..., G'_k$ are composite global transactions from $G_1, ..., G_k$ respectively. Following Proposition 1, $G'_1, ..., G'_k$ are legal transactions. Thus, S' is a serializable schedule of consistent local and composite global transactions. Therefore, S' is correct. Consequently, S is also correct. □

A global transaction G_i is readily split into a set of global subtransactions such that each global subtransaction contains all operations of G_i accessing one local site. It is clear that if a global transaction has only one global subtransaction at each local site, then all of its subtransactions are locally consistent. The process of decomposition involves the further splitting of such global subtransactions. A global transaction is *decomposable* if its decomposition ensures that each decomposed global subtransaction is locally consistent and no cyclic value-dependencies remain in the decomposed global subtransactions. Based upon this definition, the enforcement of chain-conflicting serializability on global transactions can generate MDBS-serializable schedules which are correct:

Theorem 2 *Let S be a global schedule and $\mathcal{G}$ be the set of committed and decomposable global transactions in S. If $S_\mathcal{G}$ is chain-conflicting serializable, then the local serializability of S_k (for $k=1,...,m$) implies that S is MDBS-serializable and correct.*

Proof: Since the serialization order of global subtransactions enforced by chain-conflicting serializability in a global subschedule is locally extensible, all serialization orders of global subtransactions at all local sites are consistent with the same order. Thus, S is serializable without considering the local transactions which are serialized between two decomposed global subtransactions. Thus, S is MDBS-serializable. Furthermore, the decomposition of global transactions ensures that all global subtransactions are locally consistent. By Theorem 1, S is also correct. □

In order to maintain strong correctness on global schedules, each global subtransaction must also be *locally independent*. Any two global subtransactions of a single global transaction at a local site must not exchange their data at the global level, since a local transaction which is interleaved between these two global subtransactions may update the data. For instance, in Example 4, if G_1 omits $r_{G_{11}^{(2)}}(a)$ and instead uses the value of $r_{G_{11}^{(1)}}(a)$ − \$100 as the balance of a, then $G_{11}^{(1)}$ and $G_{11}^{(2)}$ are not locally independent because of the data exchange. G_1 reads an amount that is \$100 more than the actual total of accounts a and b.

We define a global transaction as *strongly decomposable* if it is decomposable and each of its global subtransactions is locally independent. Since each locally consistent global subtransaction or local transaction always results in a globally consistent state and no data exchange between any two global subtransactions at a local site is allowed, any local or global subtransaction always see

a globally consistent state. Thus, two corollaries follow directly from Theorems 1 and 2:

Corollary 1 *If all global subtransactions are locally both consistent and independent, then MDBS-serializable global schedules are strongly correct.*

Corollary 2 *Let S be a global schedule and $\mathcal{G}$ be the set of committed and strongly decomposable global transactions in S. If $S_{\mathcal{G}}$ is chain-conflicting serializable, then the local serializability of S_k (for k=1,...,m) implies that S is MDBS-serializable and strongly correct.*

The presumption of the chain-conflicting serializability of global subschedules is crucial to Theorem 2 and corollary 2, as each individual global subtransaction is not restricted to preserve global integrity constraints. Such limitations may prove to be too restrictive for many applications. In Example 4, if the following global schedule is produced:

$S_1 : r_{G_{11}^{(1)}}(a) w_{G_{11}^{(1)}}(a) r_{L_1}(a) w_{L_1}(a) r_{L_1}(b) w_{L_1}(b) r_{G_{11}^{(2)}}(b) r_{G_{11}^{(2)}}(b) r_{G_{21}}(a),$

$S_2 : r_{G_{22}}(c) r_{G_{12}}(c) w_{G_{12}}(c),$

then, following this schedule, the serialization order of global subtransactions of G_1 and G_2 is not synchronized, and G_2 reads an amount that is \$100 less than the actual total of accounts a and c. This schedule may not be correct if there is an integrity constraint of $a + c = total$ for a data item *total*. Thus, concurrency control is still necessary for the execution of global transactions.

Some further sufficient conditions can be provided for the decomposition of global transactions. For instance, if we stipulate that any two decomposed global subtransactions at a local site cannot write simultaneously to the data items over which a global or local integrity constraint is defined, then we can guarantee that such global subtransactions are decomposable. Such conditions are useful for the systematic implementation of global transaction decomposition. However, such conditions are not necessary and may be excessively restrictive.

We have therefore shown that a global transaction can have two or more global subtransactions at a local site, provided that certain properties are satisfied. Furthermore, global serializability need not be required as a correctness criterion for concurrency control in such a scenario, and global database consistency can still be preserved.

6 Related Issues

The condition regarding the acyclicity of value-dependency relationships among global transactions has significantly relaxed the restriction on global transactions using retry approach. This relaxation may still be restrictive in some applications. For instance, consider a global transaction that reads data item a at local site LS_1 and data item b at local site LS_2, and then, depending upon the values read, updates both data items a and b. Such a global transaction would not be decomposable. In general, the GTM would be unlikely to be capable of directing two cyclic value-dependent global subtransactions either to commit sequentially or to commit simultaneously at different local sites without imposing restrictions on local sites[5] [MEKSA92]. Therefore, the global application of an MDBS would be limited, and application programmers would need to specify proper global transactions for execution in the MDBS environment.

A major issue to the implementation of the proposed approach is the enforcement of a strong decomposition of global transactions. Toward this end, the semantic information contained in global transactions must be used for the specification of decomposition. Breakpoints [FO89] are inserted in global transactions to instruct the GTM where to decompose the global transactions before submitting them to local sites. Because of the extent of semantic information involved, we assume that the specification of breakpoints is made by application programmers, rather than by the system (see also [GM83, FO89, ZJ93]).

The traditional transaction model must be modified to accommodate the specification of such breakpoints. However, as breakpoints can be removed from global transactions after global-level decomposition is complete, decomposed global subtransactions contain no breakpoints. There are therefore no special requirements made of the transaction models used at local sites, and the heterogeneity of local database systems can still be assumed.

Based upon the above proposals, global concurrency control and atomic commitment can be simplified to the question of synchronizing the serialization and commitment orders of global subtransactions at all local sites without imposing restrictions on local database systems other than the maintenance of local serializability and recoverability. Such a simplification of global transaction management permits a novel approach to decentralized global transaction management in the MDBS environment. A decentralized implementation of the proposed approaches in the MDBS

[5]The sequential commit-retry approach combining with redo and compensation techniques can further reduce this difficulty under certain conditions [BGMS92]. Such possibility will not be addressed here.

environment appears in [ZCEB93].

7 Conclusions

In this paper, we have investigated the restrictions on global transaction management necessary to maintain the isolation and atomicity 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. Our ultimate goal has been to determine the most relaxed restrictions applicable to global transaction management while maintaining the correctness of both global and local transactions.

The incorporation of chain-conflicting serializability and the sequential commit-retry approach into the unifying principle presented in this paper yields an effective approach to the problem of global transaction management. A global concurrency control correctness criterion which is less restrictive than global serializability has been developed to maintain global consistency. These approaches make feasible the implementation of a hierarchical two-level transaction management strategy in the multidatabase environment without violation of local autonomy. The limitations of global transaction management have also been explored.

References

[AA92] D. Agrawal and A. El Abbadi. Transaction management in database systems. In Ahmed K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, 1992.

[AGMS87] R. Alonso, H. Garcia-Molina, and K. Salem. Concurrency control and recovery for global procedures in federated database systems. In *IEEE Data Engineering Bulletin*, pages 5–11, September 1987.

[BGMS92] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Overview of multidatabase transaction management. *The VLDB Journal*, 1(2):181–239, October 1992.

[BGRS91] Y. Breitbart, D. Georgakopolous, M. Rusinkiewicz, and A. Silberschatz. On Rigorous

Transaction Scheduling. *IEEE Transactions on Software Engineering*, 17(9):954–960, 1991.

[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.

[BS92] Y. Breitbart and A. Silberschatz. Strong recoverability in multidatabase systems. In *Proc. of the 2nd International Workshop on Research Issues on Data Engineering : Transaction and Query Processing*, pages 170–175, Tempe, AZ, 1992.

[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.

[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.

[ED90] A. Elmagarmid and W. Du. A paradigm for concurrency control in heterogeneous distributed database systems. In *Proceedings of the Sixth International Conference on Data Engineering*, Los Angeles, California, February 1990.

[FO89] A. Farrag and T. Ozsu. Using semantic knowledge of transactions to increase concurrency. *ACM Transactions on Database Systems*, 14(4):503–525, December 1989.

[GM83] H. Garcia-Molina. Using Semantic Knowledge for Transaction Processing in a Distributed Database. *ACM Transactions on Database Systems*, 8(2):186–213, June 1983.

[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, 1988.

[GMS87] H. Garcia-Molina and K. Salem. Sagas. In *Proceedings of the ACM Conference on Management of Data*, pages 249–259, May 1987.

[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.

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

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

[LKS91] E. Levy, H. Korth, and A. Silberschatz. An optimistic commit protocol for distributed transaction management. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Denver, Colorado, May 1991.

[MEKSA92] James G. Mullen, Ahmed K. Elmagarmid, Won Kim, and Jamshid Sharif-Askary. On the impossibility of atomic commitment in multidatabase systems. In *Proceedings Of The International Conference on Systems Integration*, pages 625–634, Morristown, New Jersey, June 1992.

[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.

[MRKS91] 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.

[RAZ92] Y. RAZ. The principle of commitment ordering, or guaranteeing serilizability in a heterogeneous environment of mutiple autonomous resource-managers. (DEC-TR 841), 1992.

[RS91] Marek Rusinkiewicz and Amit Sheth. Polytransactions for managing interdepndent data. *IEEE Data Engineering Bulletin*, 14(1), March 1991.

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

[SKS91] Nandit Soparkar, Henry F. Korth, and Abraham Siberschatz. Failure-resilient transaction management in multidatabases. *IEEE Computer*, 24(12):28–36, December 1991.

[SSU91] A. Silberschatz, M. Stonebraker, and J. Ullman. Database systems: Achievements and opportunities. *Communication of ACM*, 34(10):110–120, 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 *Proc. Int'l Conf. On Data Engineering*, 1992.

[ZCEB93] Aidong Zhang, Jiansan Chen, Ahmed Elmagarmid, and Omran Bukhres. Decentralized Global Transaction Management in Multidatabase Systems. Technical Report CSD-TR-93-016, Purdue University, Feb. 1993.

[ZE93a] Aidong Zhang and Ahmed K. Elmagarmid. On global transaction scheduling criteria in multidatabase systems. In *Proceedings of the Second International Conference on Parallel and Distributed Information Systems*, pages 117–124, San Diego, California, January 1993.

[ZE93b] Aidong Zhang and Ahmed K. Elmagarmid. A theory of global concurrency control in multidatabase systems. *The VLDB Journal*, July 1993. (invited).

[ZJ93] Aidong Zhang and Jin Jing. On structural features of global transactions in multidatabase systems. In *Proceedings of the Third International Workshop on Interoperability in Multidatabase Systems*, Vienna, Austra, April 1993. (to appear).