1992

A Theory of Global Concurrency Control in Multidatabase Systems

Aidong Zhang

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Report Number:
92-098

A THEORY OF GLOBAL CONCURRENCY CONTROL IN MULTIDATABASE SYSTEMS

Aidong Zhang
Ahmed K. Elmagarmid

CSD-TR-92-098
December 1992

A Theory of Global Concurrency Control in Multidatabase Systems *

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

Abstract

This paper presents a theoretical basis for global concurrency control to maintain global serializability in multidatabase systems. Three correctness criteria are formulated to utilize the intrinsic characteristics of global transactions at the global level to determine the serialization order of global subtransactions at each local site. In particular, two new types of serializability, chain-conflicting serializability and sharing serializability, are proposed, and an optimal criterion (termed hybrid serializability) combining these two basic criteria is discussed. These criteria offer the advantage of imposing no restrictions on local sites other than local serializability while retaining global serializability. The graph testing techniques of the three criteria are provided as guidance for global transaction scheduling. In addition, the optimal aspect of hybrid serializability defines the upper limit on global serializability in multidatabase systems.

Keywords: Multidatabases, global concurrency control, chain-conflicting serializability, sharing serializability, hybrid serializability, optimality.

*Zhang is supported by an David Ross Fellowship and Elmagarmid is supported by NSF under grant IRI-8857952.

1 Introduction

Centralized databases were predominant during the decade of the nineteen-seventies, when many diverse database systems were developed based on relational, hierarchical, and network models. As applications arose that demanded increased cooperation between these systems, it became necessary to consider methods for the integration of pre-existing database systems. The design of global database systems must allow these diverse pre-existing database systems to be accessed in a unified manner while at the same time not subjecting them to conversion or major modifications. A multidatabase system (MDBS) is such a global database system.

The overriding concern of any MDBS is the preservation of local autonomy. Various aspects of local autonomy, such as design, execution, and control, have been studied in [GMK88, BS88, Pu88, Vei90], and their effects on multidatabase systems are discussed in [DEK90]. In essence, a multidatabase system may not have the ability to fully modify, control, and have knowledge of component database systems. For instance, a multidatabase system may have to deal with the heterogeneity of local database systems. This autonomy distinguishes multidatabase systems from traditional distributed database systems. Therefore, many of the early techniques developed for distributed database systems are no longer applicable to multidatabase systems. New principles and protocols need to be developed for multidatabase systems.

This paper is concerned with the issue of global transaction concurrency control. The goal of concurrency control is to ensure that transactions behave as if they are executed in isolation. Serializability[1], the conventional concurrency control correctness criterion, is adopted as the global concurrency control correctness criterion. The difficulty of maintaining global serializability in multidatabase systems has been evident in the recent literature [AGMS87, BS88, Pu88, DE89]. The integration of autonomous local database systems, each with its own concurrency controller (or scheduler), into a multidatabase via a global concurrency controller inevitably gives rise to a hierarchical structure of global concurrency control. At the lower level, local concurrency controllers maintain local serializability at local sites, while at the higher level the global concurrency controller maintains global serializability. These two levels are highly interrelated. Global subtransactions, which will be defined precisely in Section 2, are received by the local concurrency controller and treated as local transactions. The global concurrency controller, on the other hand, must reflect the serialization orders in a manner which is consistent with the local counterparts. In other words,

[1] In this paper, serializability refers to conflict serializability [Pap86].

the serialization order of global subtransactions in a local concurrency controller must somehow be reflected or inherited by the global concurrency controller. Thus, the most fundamental issue of global serializability is whether and how the global concurrency controller can determine the serialization order of global subtransactions at each local site without violation of local autonomy.

Some approaches to the above issue propose to relax global serializability theory and 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, the requirement of no value dependency among sites is allowed in quasi-serializability, and restricted Read-Write models are employed in two-level serializability. Other methods impose special restrictions on local database management systems. These approaches, such as rigorous local schedules [BGRS91] or strongly recoverable local schedules [BS92], have achieved some initial success. If the pre-existing local transaction management systems satisfy these restrictions, then these theories 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[2].

In this paper, we provide a theoretical basis for global transaction scheduling to maintain global serializability. In particular, we address the scenario in which the local databases are required only to ensure serializability. Specifically, we attempt to answer the following:

(i) What are the sufficient conditions for the global concurrency controller to determine the serialization orders of global subtransactions at local sites without imposing additional restrictions on local database systems; and

(ii) What is the weakest sufficient condition on global transaction scheduling approaches.

We shall therefore seek to determine the maximum set of globally serializable schedules that can be developed in an MDBS environment without violation of local autonomy. In general, the global concurrency controller has no information about the local serialization orders, and the execution orders of global subtransactions may 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. Thus, the key to approaching the above two questions is the avoidance of the problems caused by local indirect conflicts. This paper proposes the use of novel global scheduling criteria to achieve

[2]In [Pu88, ED90, MRB+92], an approach which utilizes the information regarding serialization events or serialization functions contained in local concurrency control protocols is proposed to solve the problem. However, such information may not be generally available.

this goal. Two basic criteria for global transaction scheduling, chain-conflicting serializability and sharing serializability, are introduced, and hybrid serializability, an optimal criterion which combines these two basic criteria, is proposed. The optimal aspect of hybrid serializability indicates the maximum class of global schedules that may be generated at the global level to maintain global serializability.

The remainder of this paper is organized as follows. Section 2 introduces the system model, defines the relevant terminology, and presents the background of the problem. Sections 3 and 4 discuss, in turn, the two basic criteria of global transaction scheduling, chain-conflicting serializability and sharing serializability. In Section 5, hybrid serializability, which combines the features of the two basic criteria, is analyzed, and its optimality is discussed. In Section 6, the present research is compared with related work and the effect of failures on the global concurrency control theory is investigated. Conclusions are set forth in Section 7.

2 Preliminaries

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

2.1 The System Model

An MDBS consists of a set of $\{LDBS_i$, for $1 \leq i \leq m\}$, where each $LDBS_i$ is a pre-existing autonomous database management system on a set of data items D_i, superimposed on which is a global database management system (GDBS). Figure 1 depicts the model.

Global transactions (G) are submitted to the GDBS and then divided into a set of global subtransactions which are submitted to the LDBSs individually, while local transactions (L) are directly submitted to LDBSs. Furthermore, as stated in [GPZ86], global serializability cannot be generally maintained in MDBSs if a global transaction has more than one subtransaction at a given local site. Thus, we assume that each global transaction has at most one subtransaction at each local site.

As a necessary assumption of global serializability, we also presume that the concurrency control mechanisms of LDBSs ensure local serializability. However, no restriction is imposed on these mechanisms.

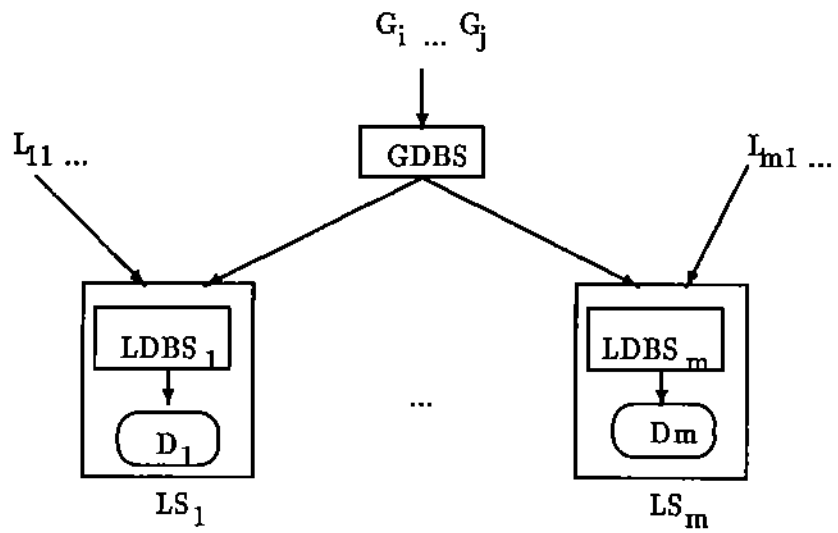


Figure 1: Conceptual multidatabase architecture

2.2 Notations 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 *share* with each other if they access the same data item. Two operations *conflict* with each other if they are sharing operations 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 contain exactly one termination operation that is the maximum (last) element in the partial order. A more formal definition of a transaction can be found in [BHG87, Had88]. A local transaction L_{ij} is a transaction that accesses the data items at a single local site LS_i. 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. The global transaction G_i consists of a set of global subtransactions $\{G_{ij_1}, G_{ij_2}, \cdots, G_{ij_r}\}$ where the subtransaction G_{ij_l} $(1 \leq l \leq r)$ is a transaction accessing $LDBS_{j_l}$. A set $\mathcal{G} = \{G_1, \cdots, G_n\}$ contains those global transactions that are submitted to the GDBS, and $\mathcal{G}_k$ denotes the set of global subtransactions of $\mathcal{G}$ at local site LS_k. A transaction T_i refers to either a local or global transaction, and OP_{T_i} denotes the set of operations contained in T_i.

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 $o_j \in OP_{T_j}$.

A schedule over a set of transactions is a partial order of all and only the operations of these transactions which orders all conflicting operations and 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. A global subschedule $S_\mathcal{G}$ is S restricted to the set $\mathcal{G}$ 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 the schedule s if all operations of T_1 are executed before any operation of T_2 in s. Obviously, a total execution order on transactions in a serial schedule can be determined. We denote $o_1 \prec^s_{eo} o_2$ if operation o_1 is executed before operation o_2 in schedule s. We denote $T_1 \prec^s_{eo} T_2$ if, for transactions T_1 and T_2 in s and every operation $o_1 \in T_1$ and every operation $o_2 \in T_2$, $o_1 \prec^s_{eo} o_2$.

Let s be a schedule and $C(s)$ be s restricted to the committed transactions in s. We say that s is serializable if there exists a serial schedule s' and $C(s)$ is (conflict) equivalent[3] to s'. The execution order of transactions in s' is a serialization order of s. Thus, a global schedule S is serializable if and only if S is serializable in a total order on both committed global and local transactions in S. We denote $T_1 \prec^s_{sr} T_2$ if T_1 precedes T_2 in the serialization order of s.

2.3 Global Serialization Theorem

Since a global schedule is the combination of all local schedules, the global serialization order must inherit local serialization orders. On the other hand, the relative serialization order of the global subtransactions of each global transaction at all local sites needs to be synchronized to maintain global serializability [BS88].

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 same order symbol as that of the global transaction to which it belongs. The following theorem 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

[3]See the definition given in [BHG87, Had88].

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 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.*

Theorem 1 has been identified in [MRB+92]; its proof is given in Appendix A.

The above theorem shows that the maintenance of global serializability can be reduced to synchronizing the relative serialization orders of global subtransactions of each global transaction at all local sites. This further implies that the serializability of local schedules, on their own, is not sufficient to maintain global serializability, since global subtransactions in different local databases may have different serialization orders.

Though Theorem 1 provides a necessary and sufficient condition to maintain global serializability, due to the constraints of local autonomy, the GDBS may not be able to generate all global schedules satisfying this condition. Our research has sought to identify alternative correctness conditions to be placed on global subschedules to provide sufficient conditions for the GDBS to maintain global serializability without imposing restrictions on local sites.

2.4 Effects of Local Indirect Conflicts

In their early work [GPZ86], Gligor and Popescu-Zeletin considered it sufficient to synchronize the serialization orders of global subtransactions which conflict at local sites. It was generally believed that non-conflict global subtransactions had no effect on global serializability. Later results reported in [BS88, DE89] indicated that, due to local indirect conflicts, the execution order of global subtransactions at a local site may not be consistent with their serialization order, even if they do not conflict. The following example 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:*

$G_1 : w_{g_1}(a) r_{g_1}(b)$

$G_2 : r_{g_2}(a) w_{g_2}(c)$

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 in 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, arising 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 $r_{g_1}(b)$).

Thus, even though the execution orders of the global subtransactions at all local sites are 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 factor in the difficulty of 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 GDBS has no knowledge of the submissions of local transactions.

This discussion of local indirect conflicts indicates how the characteristics of local transactions determine the serialization order of global subtransactions at local sites. Conversely, we observe that the characteristics of global transactions can also indirectly effect the serialization order of local schedules at local sites. For instance, if, in Example 1, 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 existence of conflict between global subtransactions G_{12} and G_{22} here imposes an indirect effect on local scheduling. As another instance, if, in Example 1, G_2 is instead defined as $r_{g_2}(a)r_{g_2}(b)$ and the execution of $r_{g_1}(b)$ at site LS_2 precedes the execution of $r_{g_2}(b)$, then $G_1 \prec_{sr}^{S_2} G_2$ will always be assured in LS_2 (note that $G_2 \prec_{sr}^{S_2} G_1$ may be simultaneously true), even though G_{12} and G_{22} do not conflict. This is due to the fact that there is no local transaction L which can conflict with G_{12} and G_{22} such that $G_2 \prec_{sr}^{S_2} L \prec_{sr}^{S_2} G_1$. We will discuss these properties in detail in the next two sections.

3 Chain-Conflicting Serializability

In this section, we investigate a correctness criterion on global subschedules which maintains the execution order of conflicting operations of global subtransactions to be identical to the serialization order of the global subtransactions at each local site. This criterion, termed chain-conflicting serializability, provides a sufficient condition for the GDBS to synchronize the relative serialization orders of the global subtransactions of each global transaction at all local sites without imposing any restrictions other than requiring each *LDBS* to ensure local serializability.

3.1 The Principle

Definitions of chain-conflicting transactions and chain-conflicting serializable schedules will first be provided. We will then show that, if global subschedules are chain-conflicting serializable, global serializability is assured. No restriction other than local serializability is required at local sites.

Definition 1 (Chain-conflicting transactions) *A set T of local transactions is chain-conflicting if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $T_{i_1} \overset{c}{\sim} T_{i_2} \overset{c}{\sim} \cdots \overset{c}{\sim} T_{i_n}$. A set $\mathcal{G}$ of global transactions is chain-conflicting if there is a total order O on $\mathcal{G}$ such that for all k,$1 \leq k \leq m$, $\mathcal{G}_k$ is chain-conflicting in an order consistent with O.*

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. Three global transactions are given as follows:*

$G_1 : r_{g_1}(a) w_{g_1}(b) r_{g_1}(c)$

$G_2 : w_{g_2}(a)$

$G_3 : r_{g_3}(a) r_{g_3}(b)$

where $\{G_1, G_2, G_3\}$ is chain-conflicting in the order $G_1 \rightarrow G_2 \rightarrow G_3$. An alternative chain-conflicting order is $G_3 \rightarrow G_2 \rightarrow G_1$. No other chain-conflicting orders exist. Note that G_2 does not have a global subtransaction at local site LS_2.

Note that $T_1 \overset{c}{\sim} T_2 \overset{c}{\sim} T_3$ may not imply $T_1 \overset{c}{\sim} T_3$ and that a set of transactions in which all transactions are in mutual conflict is always chain-conflicting in any order.

Definition 2 (Chain-conflicting serializability) *A schedule s is chain-conflicting serializable if the set T of committed transactions in s is chain-conflicting in a total order O on T and s is serializable in O.*

Definition 2 implies that chain-conflicting serializability is stronger than serializability; in other words, chain-conflicting serializability implies serializability.

We will now illustrate the application of chain-conflicting serializability in an MDBS environment. We give the following main theorem first.

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

The proof of this theorem relies on Lemma 1, which shows that the outcome of a concurrent execution of transactions depends only on the relative ordering of conflicting operations [BHG87].

Lemma 1 *If o_1 and o_2 are conflicting operations of transactions T_1 and T_2 (respectively) in a serializable schedule s, then $o_1 \prec^s_{co} o_2$ if and only if $T_1 \prec^s_{sr} T_2$.*

Proof: (*if*) We need to show that $T_1 \prec^s_{sr} T_2$ implies $o_1 \prec^s_{co} o_2$. Suppose $o_1 \not\prec^s_{co} o_2$. Then, since o_1 and o_2 conflict, we must have $o_2 \prec^s_{co} o_1$. Thus, in any serial schedule s' which is conflict equivalent to S, $o_2 \prec^{s'}_{co} o_1$. Hence, $T_1 \not\prec^s_{sr} T_2$.

(*only if*) Conversely, we need to show that $o_1 \prec^s_{co} o_2$ implies $T_1 \prec^s_{sr} T_2$. Similarly to above, suppose $T_1 \not\prec^s_{sr} T_2$. Then, since s is serializable, we must have $T_2 \prec^s_{sr} T_1$. Since o_1 conflicts o_2, in any serial schedule s' which is conflict equivalent to s, $T_2 \prec^{s'}_{co} T_1$, which implies $o_2 \prec^{s'}_{co} o_1$. Hence, $o_2 \prec^s_{co} o_1$. Consequently, $o_1 \not\prec^s_{co} o_2$. □

We now apply Lemma 1 to the MDBS environment. Assume a global subschedule $S_{\mathcal{G}}$ of global schedule S is serializable in a total order O on $\mathcal{G}$, and $G_i \in \mathcal{G}$ precedes $G_j \in \mathcal{G}$ in O. If, for integer k ($1 \leq k \leq m$), $G_{ik} \overset{c}{\sim} G_{jk}$ and o_{ik}, o_{jk} are conflicting operations of G_{ik} and G_{jk}, respectively, then, by the *"if"* part of Lemma 1, $o_{ik} \prec^{S_{\mathcal{G}}}_{co} o_{jk}$. Consequently, at local site LS_k, $o_{ik} \prec^{S_k}_{co} o_{jk}$. If S_k is serializable, then, by the *"only if"* part of Lemma 1, $G_{ik} \prec^{S_k}_{sr} G_{jk}$. We have shown that the conflicting characteristics of global transactions can indirectly affect the serialization orders of global subtransactions in local schedules.

We now present the proof of Theorem 2.

Proof: Suppose $S_{\mathcal{G}}$ is chain-conflicting serializable in a total order $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$ on $\mathcal{G}$. Without loss of generality, we assume that, at local site LS_k ($1 \leq k \leq m$), $G_{i_1 k}, G_{i_2 k}, \cdots, G_{i_n k}$ exist. We need to prove that, if S_k is serializable, then $G_{i_1 k} \prec^{S_k}_{sr} G_{i_2 k} \prec^{S_k}_{sr} \cdots \prec^{S_k}_{sr} G_{i_n k}$. The proof proceeds by induction on integer n:

n=1: Straightforward.

Suppose for $n = j(> 1)$, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_jk}$ holds.

Consider $n = j+1$. Since G_{i_j} precedes $G_{i_{j+1}}$ in O, $G_{i_jk} \overset{c}{\sim} G_{i_{j+1}k}$. If o_{i_jk} and $o_{i_{j+1}k}$ are conflicting operations of G_{i_jk} and $G_{i_{j+1}k}$, respectively, then, by the *"if"* part of Lemma 1, $o_{i_jk} \prec_{co}^{S_G} o_{i_{j+1}k}$, which is equivalent to $o_{i_jk} \prec_{co}^{S_k} o_{i_{j+1}k}$. Then, by the *"only if"* part of Lemma 1, $G_{i_jk} \prec_{sr}^{S_k} G_{j+1k}$.

Thus, our induction proof shows $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$. Hence, the serialization order of global subtransactions in S_k $(1 \leq k \leq m)$ is consistent with O. Consequently, by Theorem 1, S is serializable. □

The fundamental concern of chain-conflicting serializability is to formulate the weakest conflicting relationship on global transactions such that the GDBS can indirectly determine the serialization order of global subtransactions at local sites without violation of local autonomy. We shall address this issue more precisely in Section 5.

3.2 Graph Testing of Chain-conflicting Serializability

Following Theorem 2, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is chain-conflicting. In addition, only conflicting operations need be ordered. A traditional graph-theoretic characterization of chain-conflicting serializability for global transaction execution ordering is discussed below.

Let us first introduce the global transaction execution graph.

Definition 3 *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S, $\mathcal{G}$ being chain-conflicting in a total order O on $\mathcal{G}$. The global transaction execution graph of $S_\mathcal{G}$ in O, denoted $GEG_c(S_\mathcal{G}^O)$, is a directed graph whose nodes are the global transactions in $\mathcal{G}$ and whose edges are all the relations $(G_i, G_j)(i \neq j)$ such that $G_i \rightarrow G_j$ if and only if : (1) G_i precedes G_j in O; or (2) at LS_k $(1 \leq k \leq m)$, there exist conflicting operations $o_{ik} \in OP_{G_{ik}}$, $o_{jk} \in OP_{G_{jk}}$ and $o_{ik} \prec_{co}^{S_k} o_{jk}$.*

Theorem 3 (Global execution theorem) *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S. If $\mathcal{G}$ is chain-conflicting in a total order O on $\mathcal{G}$, then $S_\mathcal{G}$ is chain-conflicting serializable in O if and only if $GEG_c(S_\mathcal{G}^O)$ is acyclic.*

Proof : Let $S = \{S_1, S_2, ..., S_m\}$ be a global schedule and $\mathcal{G}$ be the set of committed global

transactions in S, and $\mathcal{G}$ being chain-conflicting in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$.

(*if*) Since $GEG_c(S_{\mathcal{G}}^O)$ is acyclic, it can be topologically sorted. Obviously, by the definition of $GEG_c(S_{\mathcal{G}}^O)$, $G_{i_1}, G_{i_2}, ..., G_{i_n}$ must be the topological sort of $GEG_c(S_{\mathcal{G}}^O)$. Let $S'_{\mathcal{G}}$ be the serial schedule $G_{i_1}, G_{i_2}, ..., G_{i_n}$. We claim that $S_{\mathcal{G}}$ is conflict equivalent to $S'_{\mathcal{G}}$. To illustrate this, let $op_i \in OP_{G_i}$ and $op_j \in OP_{G_j}$, where G_i, G_j are committed global transactions in S. Suppose op_i and op_j conflict and $op_i \prec_{eo}^{S_{\mathcal{G}}} op_j$. By the definition of $GEG_c(S_{\mathcal{G}}^O)$, $G_i \rightarrow G_j$ is an edge in $GEG_c(S_{\mathcal{G}}^O)$. Thus, in $S'_{\mathcal{G}}$, all operations of G_i appear before any operation of G_j, and in particular, $op_i \prec_{eo}^{S'_{\mathcal{G}}} op_j$. In a situation comparable to the proof of the serialization theorem in [BHG87], $S_{\mathcal{G}}$ is conflict equivalent to $S'_{\mathcal{G}}$. Hence, $S_{\mathcal{G}}$ is chain-conflicting serializable in O.

(*only if*) Let $S_{\mathcal{G}}$ be chain-conflicting serializable in O. Let $S'_{\mathcal{G}}$ be a serial schedule $G_{i_1} G_{i_2} ... G_{i_n}$ which is conflict equivalent to $S_{\mathcal{G}}$. Consider an edge $G_i \rightarrow G_j$ in $GEG_c(S_{\mathcal{G}}^O)$. Either G_i precedes G_j in O or there are two conflicting operations o_i,o_j of G_i,G_j (respectively) such that $o_i \prec_{eo}^{S_{\mathcal{G}}} o_j$. Thus, it follows that G_i appears before G_j in $S'_{\mathcal{G}}$, since $S'_{\mathcal{G}}$ is serial in O and conflict equivalent to $S_{\mathcal{G}}$. Let there be a cycle in $GEG_c(S_{\mathcal{G}}^O)$ which, without loss of generality, is $G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_r \rightarrow G_1$ ($r > 1$). These edges imply that, in $S'_{\mathcal{G}}$, G_1 appears before G_2 which appears before G_3 which appears ... before G_r which appears before G_1. Thus, the existence of the cycle implies that each of $G_1, G_2, \cdots, G_r$ appears before itself in the serial schedule $S'_{\mathcal{G}}$, thus contradicting our assumption. Hence, $GEG_c(S_{\mathcal{G}}^O)$ is acyclic. □

A sufficient condition for global transaction scheduling to maintain global serializability in a failure-free multidatabase environment follows directly from Definition 3 and Theorem 3. That is, the execution order of conflicting operations of global transactions must act in accordance with the order of their chain-conflicting aspects. This condition is applicable to the global transaction concurrency controller since, as we have indicated in the system model, the GDBS can control the submissions of global transactions. Consequently, the execution order of global transactions can be controlled at global level. We give an illustrative example below. The enforcement of chain-conflicts on global transactions will be discussed in the next subsection, and the effect of failures on global concurrency control will be discussed in Section 6.2.

Example 3 *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:*

$G_1 : w_{g_1}(a) r_{g_1}(b)$

$G_2 : r_{g_2}(a) w_{g_2}(c) w_{g_2}(b)$

$G_3 : w_{g_3}(a) r_{g_3}(c)$

which is chain-conflicting in the order $G_1 \rightarrow G_2 \rightarrow G_3$. *Let* L_{21} *be a local transaction submitted at local site* LS_2:

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

Let $S = \{S_1, S_2\}$ *be the global schedule:*

$S_1 : w_{g_1}(a) r_{g_2}(a) w_{g_3}(a)$

$S_2 : w_{L_{21}}(b) r_{g_1}(b) w_{L_{21}}(c) w_{g_2}(c) r_{g_3}(c) w_{g_2}(b)$.

Obviously, S_Q *is chain-conflicting serializable in the order* $G_1 \rightarrow G_2 \rightarrow G_3$, *and* S *is serializable. Note that, as long as the execution orders of conflicting operations of global subtransactions are controlled identically at both local sites, such as:*

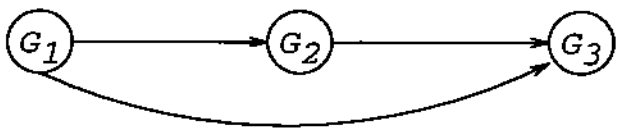
$$w_{g_1}(a) \prec_{co}^{S_1} r_{g_2}(a) \prec_{co}^{S_1} w_{g_3}(a)$$

$$r_{g_1}(b) \prec_{co}^{S_2} w_{g_2}(b)$$

$$w_{g_2}(c) \prec_{co}^{S_2} r_{g_3}(c)$$

then global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems.

In $GEG_c(S^O_{\{G_1,G_2,G_3\}})$, *we have:*



Note that $G_{12} \overset{c}{\not\rightarrow} G_{32}$. *In the following schedule* S':

$S'_1 : w_{g_1}(a) w_{g_3}(a) r_{g_2}(a)$,

$S'_2 : w_{L_{21}}(b) r_{g_1}(b) r_{g_3}(c) w_{L_{21}}(c) w_{g_2}(c) w_{g_2}(b)$,

S'_Q *is serializable (not chain-conflicting serializable) in the order* $G_1 \rightarrow G_3 \rightarrow G_2$, *but* S' *is not serializable.* □

3.3 Forcing Chain-Conflicts in Global Transactions

One advantage of chain-conflicting serializability is that it can be easily generalized to all global transactions by forcing chain-conflicts in global transactions. For example, an elegant method, termed the ticket method, is proposed in [GRS91]. The ticket method introduces a data item

called *ticket* at each local site and requires each global subtransaction to access the ticket at its site. Consequently, conflicts are created among all global subtransactions which are executed at the same site. The ticket method thus generates an instance which satisfies a strong condition of the chain-conflicting property; that is, tickets cause the set of all global transactions to be chain-conflicting in any order. A minor problem with the ticket method is that a local site may not be willing to allow the creation of a ticket in its database.

An alternative method, which we will term the extra operation method, may be suggested to circumvent this difficulty. Let G_{ik} and G_{jk} be global subtransactions in local site LS_k which do not conflict. Chain-conflicts can then be simulated. Suppose G_{ik} is executed before G_{jk}. If G_{ik} and G_{jk} have no conflict and the last operation of G_{ik} is on data item x, then we append operations $r(x)$ and $w(x)$ to G_{jk}. Let G'_{jk} denote G_{jk} after appending these extra operations. Now G_{ik} and G'_{jk} 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 implementation of this method can be transparent to application programmers; that is, the global concurrency controller can hide the details of implementing the enforcement of chain-conflicts from application programmers.

The degree of difficulty of enforcing chain-conflicts on global transactions varies with the available interface between the GDBS and the LDBSs. Most current research assumes the availability of global transaction operations submitted by the GDBS to the LDBSs and that the completion of these operations is acknowledged by the LDBSs to the GDBS [MRB+92, BGMS92]. In such cases, the extra operation method can certainly be implemented. In Section 5, it will be seen that the insertion of update operations may be avoided.

4 Sharing Serializability

In this section, we investigate another correctness criterion of global subschedules, one which maintains that the execution order of the sharing operations of global subtransactions is identical to their serialization order at each local site. This criterion, termed sharing serializability, provides another sufficient condition for the GDBS to synchronize the relative serialization orders of the global subtransactions of each global transaction at all local sites.

4.1 The Principle

The definitions of fully sharing transactions and sharing serializable schedules will first be provided. We will then show that, if global subschedules are sharing serializable, global serializability is assured. No restriction other than local serializability is required at local sites.

Let D_T denote the set of data items that transaction T accesses.

Definition 4 (Fully sharing transactions) *A set T of local transactions is fully sharing if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $D_{T_{i_1}} \subseteq D_{T_{i_2}} \subseteq \cdots \subseteq D_{T_{i_n}}$. A set $\mathcal{G}$ of global transactions is fully sharing if there is a total order O on $\mathcal{G}$ such that for all k,$1 \leq k \leq m$, G_k is fully sharing in an order consistent with O.*

The fully sharing relationship of transactions is defined with respect to all data accessed by transactions exclusive of types of operations. A set of transactions may be chain-conflicting but not fully sharing, or it may be fully sharing but not chain-conflicting. In Example 2, $\{G_1,G_2,G_3\}$ is fully sharing in the order $G_2 \rightarrow G_3 \rightarrow G_1$. There is no other alternative fully sharing relation.

The execution order of sharing operations of transactions can also determine the serialization order of the transactions, as expressed in the following lemma:

Lemma 2 *Assume that T_1 and T_2 are transactions in a serializable schedule s such that $D_{T_1} \subseteq D_{T_2}$. If, for all sharing operations $o_1 \in OP_{T_1}, o_2 \in OP_{T_2}$, $o_1 \prec^s_{co} o_2$, then $T_1 \prec^s_{sr} T_2$.*

Proof: (1) If T_1 and T_2 conflict, then since conflicting operations must access common data, there exist conflicting operations $o_1 \in OP_{T_1}, o_2 \in OP_{T_2}$, $o_1 \prec^s_{co} o_2$. Hence, $T_1 \prec^s_{sr} T_2$ follows from Lemma 1; otherwise

(2) If T_1 and T_2 do not conflict, then we need to prove that there is no transaction T' which conflicts with T_1, and consequently also conflicts with T_2 (since $D_{T_1} \subseteq D_{T_2}$), such that $T_2 \prec^s_{sr} T' \prec^s_{sr} T_1$.

The proof proceeds by contradiction. Suppose we do have a transaction T' which conflicts with T_1 and T_2 such that $T_2 \prec^s_{sr} T' \prec^s_{sr} T_1$. Since $D_{T_1} \subseteq D_{T_2}$, an operation of T' which conflicts with T_1 must also conflict with T_2. Without loss of generality, let o_1, o', and o_2 be conflicting operations of T_1, T', and T_2, respectively. By Lemma 1, we have $o_2 \prec^s_{co} o' \prec^s_{co} o_1$, contradicting the assumption $o_1 \prec^s_{co} o_2$. □

We shall now introduce sharing serializability.

Definition 5 (Sharing equivalence) *Two global subschedules $S_{\mathcal{G}}$ and $S'_{\mathcal{G}}$ of global schedule S and S' are said to be sharing equivalent, denoted $S_{\mathcal{G}} \equiv_s S'_{\mathcal{G}}$, if they have the same operations of $\mathcal{G}$, where $\mathcal{G}$ is fully sharing in a total order O on $\mathcal{G}$ and if G_i precedes G_j in O, then for each integer k $(1 \leq k \leq m)$ and all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_{\mathcal{G}}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_{\mathcal{G}}} o_{jk}$.*

Definition 6 (Sharing serializability) *A global subschedule $S_{\mathcal{G}}$ is sharing serializable if and only if $C(S_{\mathcal{G}})$ is sharing equivalent to a serial global subschedule.*

Note that sharing serializability is stronger than serializability; that is, sharing serializability implies serializability.

In Example 2, a global subschedule $S_{\mathcal{G}} = w_{g_2}(a)r_{g_3}(a)r_{g_1}(a)r_{g_3}(b)r_{g_1}(b)r_{g_1}(c)$ is sharing serializable in the order $G_2 \rightarrow G_3 \rightarrow G_1$.

We shall now illustrate the application of sharing serializability in an MDBS environment, addressing first the the application of Lemma 2.

Assume a global subschedule $S_{\mathcal{G}}$ is sharing serializable in a total order O on $\mathcal{G}$, and $G_i \in \mathcal{G}$ precedes $G_j \in \mathcal{G}$ in O. If, for integer k $(1 \leq k \leq m)$, for all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_{\mathcal{G}}} o_{jk}$, then at local site LS_k, $o_{ik} \prec_{eo}^{S_k} o_{jk}$. If S_k is serializable, by Lemma 2, $G_{ik} \prec_{sr}^{S_{L_k}} G_{jk}$. We have shown that the sharing characteristics of global transactions can indirectly affect the serialization order of global subtransactions in local schedules.

Our major theorem is the following:

Theorem 4 *Let S be a global schedule and $\mathcal{G}$ be the set of global transactions in S. If $S_{\mathcal{G}}$ is sharing serializable, then the local serializability of S_k (for k=1,...,m) implies the global serializability of S*[4].

Proof: Suppose $S_{\mathcal{G}}$ is sharing serializable in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$ on $\mathcal{G}$. Without loss of generality, we assume that, at local site LS_k $(1 \leq k \leq m)$, $G_{i_1k}, G_{i_2k}, \cdots, G_{i_nk}$ exist. We need to prove that, if S_k is serializable, then $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$. The proof proceeds by induction on integer n:

n=1: Straightforward.

Suppose for $n = j(> 1)$, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_jk}$ holds.

Consider $n = j + 1$. Since G_{i_j} precedes $G_{i_{j+1}}$ in O, then for all sharing operations $o_{i_jk} \in OP_{G_{i_jk}}, o_{i_{j+1}k} \in OP_{G_{i_{j+1}k}}$, $o_{i_jk} \prec_{eo}^{S_{\mathcal{G}}} o_{i_{j+1}k}$, which is equivalent to $o_{i_jk} \prec_{eo}^{S_k} o_{i_{j+1}k}$. By Lemma 2, $G_{i_jk} \prec_{sr}^{S_k} G_{j+1k}$.

[4] A similar theory can be propounded using the relationship $D_{T_{i_1}} \supseteq D_{T_{i_2}} \supseteq \cdots \supseteq D_{T_{i_n}}$.

Thus, our induction proof shows $G_{i_1 k} \prec_{sr}^{S_k} G_{i_2 k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_n k}$. Hence, the serialization order of global subtransactions in S_k $(1 \leq k \leq m)$ is consistent with O. Consequently, by Theorem 1, S is serializable. □

The fundamental concern of sharing serializability is to seek alternative properties of global transactions other than conflicts such that the GDBS can indirectly determine the serilization order of global subtransactions at local sites without violation of local autonomy. The feasibility of this approach is to be further explored.

4.2 Graph Testing of Sharing Serializability

Following Theorem 4, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is fully sharing. In addition, only sharing operations need be ordered. This criterion shows that the serialization order of global subtransactions at a local site can be determined at the global level without requiring that the global subtransactions be conflicting. Note that both classes of global subschedules that satisfy chain conflicting serialization or sharing serializability are not disjoint.

A traditional graph-theoretic characterization of sharing serializability for global transaction execution ordering is discussed below.

Let us first introduce the global transaction execution graph.

Definition 7 *Let $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ be committed global transactions in the global schedule S, with $\mathcal{G}$ being sharing serializable in a total order O on $\mathcal{G}$. The global transaction execution graph of $S_\mathcal{G}$ in O, denoted $GEG_s(S_\mathcal{G}^O)$, is a directed graph whose nodes are the global transactions in S and whose edges are all the relations $(G_i, G_j)(i \neq j)$ such that $G_i \rightarrow G_j$ if and only if : (1) G_i precedes G_j in O; or (2) at LS_k $(1 \leq k \leq m)$, there exist sharing operations $o_{ik} \in OP_{G_{ik}}$, $o_{jk} \in OP_{G_{jk}}$ and $o_{ik} \prec_{co}^{S_k} o_{jk}$.*

Theorem 5 (Global execution theorem) *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S. If $\mathcal{G}$ is fully sharing in a total order O on $\mathcal{G}$, then $S_\mathcal{G}$ is sharing serializable in O if and only if $GEG_s(S_\mathcal{G}^O)$ is acyclic.*

Proof: Let $S = \{S_1, S_2, ..., S_m\}$ be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S, with $\mathcal{G}$ being fully sharing in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$.

(*if*) Since $GEG_s(S_\mathcal{G}^O)$ is acyclic, it can be topologically sorted. Obviously, by the definition of $GEG_s(S_\mathcal{G}^O)$, $G_{i_1}, G_{i_2}, ..., G_{i_n}$ must be the topological sort of $GEG_s(S_\mathcal{G}^O)$. Let $S'_\mathcal{G}$ be the serial schedule $G_{i_1}, G_{i_2}, ..., G_{i_n}$. We claim that $S_\mathcal{G}$ is sharing equivalent to $S'_\mathcal{G}$. To illustrate this, let $op_i \in OP_{G_i}$ and $op_j \in OP_{G_j}$, where G_i, G_j are committed global transactions in S. Suppose op_i and op_j share with each other and $op_i \prec_{eo}^{S_\mathcal{G}} op_j$. By the definition of $GEG_s(S_\mathcal{G}^O)$, $G_i \rightarrow G_j$ is an edge in $GEG_s(S_\mathcal{G}^O)$. Thus, in $S'_\mathcal{G}$, all operations of G_i appear before any operation of G_j, and in particular, $op_i \prec_{eo}^{S'_\mathcal{G}} op_j$. By Definition 5, $S_\mathcal{G}$ is sharing equivalent to $S'_\mathcal{G}$. Hence, $S_\mathcal{G}$ is sharing serializable in O.

(*only if*) Let $S_\mathcal{G}$ be sharing serializable in O. Let $S'_\mathcal{G}$ be a serial schedule $G_{i_1}G_{i_2}...G_{i_n}$ which is sharing equivalent to $S_\mathcal{G}$. Consider an edge $G_i \rightarrow G_j$ in $GEG_s(S_\mathcal{G}^O)$. Either G_i precedes G_j in O or there are two sharing operations o_i,o_j of G_i,G_j (respectively) such that $o_i \prec_{eo}^{S_\mathcal{G}} o_j$. Thus, it follows that G_i appears before G_j in $S'_\mathcal{G}$, since $S'_\mathcal{G}$ is serial in O and sharing equivalent to $S_\mathcal{G}$. Let there be a cycle in $GEG_s(S_\mathcal{G}^O)$ which, without loss of generality, is $G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_r \rightarrow G_1$ ($r > 1$). These edges imply that, in $S'_\mathcal{G}$, G_1 appears before G_2 which appears before G_3 which appears ... before G_r which appears before G_1. Thus, the existence of the cycle implies that each of $G_1, G_2, \cdots, G_r$ appears before itself in the serial schedule $S'_\mathcal{G}$, thus contradicting our assumption. Hence, $GEG_s(S_\mathcal{G}^O)$ is acyclic. □

Similarly, a sufficient condition for global transaction scheduling to maintain global serializability in a failure-free multidatabase environment follows directly from Definition 7 and Theorem 5. That is, the execution order of sharing operations of global transactions must act in accordance with the order of their fully sharing property. The following example illustrates this result.

Example 4 *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:*

$G_1 : w_{g_1}(a) r_{g_1}(b)$

$G_2 : r_{g_2}(a) w_{g_2}(c) r_{g_2}(b)$,

which is fully sharing in the order $G_1 \rightarrow G_2$. Let L_{21} be a local transaction submitted at local site LS_2:

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

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : w_{g_1}(a) r_{g_2}(a)$,

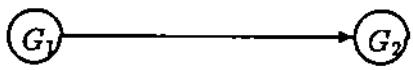
$S_2 : w_{L_{21}}(b) r_{g_1}(b) r_{L_{21}}(c) w_{g_2}(c) r_{g_2}(b)$.

Obviously, S_G is sharing serializable in the order $G_1 \rightarrow G_2$, and S is serializable. Note that G_{12} and G_{22} do not conflict. However, as long as the execution orders of sharing operations of global subtransactions are controlled in the order:

$$w_{g_1}(a) \prec_{co}^{S_1} r_{g_2}(a)$$

$$r_{g_1}(b) \prec_{co}^{S_2} r_{g_2}(b)$$

then the global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems. In $GEG_s(S^O_{\{G_1,G_2\}})$, we have:



4.3 Forcing Sharing Operations in Global Transactions

The extra operation method can also be utilized to enforce the fully sharing property on all global transactions, requiring only the insertion of retrieval operations. In this regard, sharing serializability is simpler and more efficient than chain-conflicting serializability. Though the application of the fully sharing property to global transactions may sometimes burden them with long appendices, these will always be finite, since the data items in a local database are finite. In the next section, we will show that such exponentially increasing appendices can be reduced automatically when the fully sharing property is merged with the chain-conflicting property. Nevertheless, more elegant approaches need to be investigated. At this point, use of the fully sharing property alone does not appear to offer the GDBS significant assistance toward the preservation of global serializability.

5 Hybrid Serializability

We shall now discuss hybrid serializability, a correctness criterion which exhibits characteristics of both chain-conflicting and sharing serializability. The application of the hybrid property to global transactions offers an optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing restrictions on or requiring any information from that local site other than local serializability.

5.1 Hybrid Serializability

The definitions of hybrid transactions and hybrid serializable schedules, presented below, clarify the manner in which they effectively combine the best features of chain-conflicting serializability and sharing serializability.

Definition 8 (Hybrid transactions) *A set T of local transactions is hybrid if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $T_{i_1} \Diamond T_{i_2} \Diamond \cdots \Diamond T_{i_n}$ where $\Diamond \in \{\overset{c}{\sim}, \subseteq, \supseteq\}$[5]. A set $\mathcal{G}$ of global transactions is hybrid if there is a total order O on $\mathcal{G}$ such that for all k,$1 \leq k \leq m$, $\mathcal{G}_k$ is hybrid in an order consistent with O.*

The operations that determine the hybrid property on global transactions are termed hybrid operations.

Definition 9 (Hybrid equivalence) *Two global subschedules $S_\mathcal{G}$ and $S'_\mathcal{G}$ of global schedule S and S' are said to be hybrid equivalent, denoted $S_\mathcal{G} \equiv_h S'_\mathcal{G}$, if they have the same operations of $\mathcal{G}$, where $\mathcal{G}$ is hybrid in a total order O on $\mathcal{G}$ and, for any G_i preceding G_j in O, the following conditions are satisfied for all integer k $(1 \leq k \leq m)$:*

- *if $G_{ik} \overset{c}{\sim} G_{jk}$ in O, then for all conflicting operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_\mathcal{G}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_\mathcal{G}} o_{jk}$; or*
- *if $G_{ik} \subseteq G_{jk}$ (or $G_{ik} \supseteq G_{jk}$) in O, then for all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_\mathcal{G}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_\mathcal{G}} o_{jk}$.*

Definition 10 (Hybrid serializability) *A global subschedule $S_\mathcal{G}$ is hybrid serializable if and only if $C(S_\mathcal{G})$ is hybrid equivalent to a serial global subschedule.*

Following the properties of chain-conflicting and sharing serializability, hybrid serializability is stronger than serializability; in other words, hybrid serializability implies serializability.

We will now illustrate the application of hybrid serializability in an MDBS environment.

Theorem 6 *Let S be a global schedule and $\mathcal{G}$ be the set of global transactions in S. If $S_\mathcal{G}$ is hybrid serializable, then the local serializability of S_k (for k=1,...,m) implies the global serializability of S.*

[5]We consider that $\overset{c}{\sim}$ has a higher priority to be chosen than $\subseteq$ (or $\supseteq$). That is, if two transactions T_i and T_j have both $T_i \overset{c}{\sim} T_j$ and $T_i \subseteq$ (or $\supseteq$) T_j properties, then $T_i \overset{c}{\sim} T_j$ will be chosen in the hybrid ordering instead of $T_i \subseteq$ (or $\supseteq$) T_j. Both $\subseteq$ and $\supseteq$ have the same priority.

The proof of this theorem is comparable to that of Theorem 2 and 4 and is therefore omitted here.

The fundamental thrust of hybrid serializability is the illustration of mixed features of transactions to maintain global serializability. This formulation of hybrid serializability possesses several novel features which will be discussed in the following subsections.

Following Theorem 6, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is hybrid. In addition, only hybrid operations need be ordered.

A global transaction execution graph of $S_{\mathcal{G}}$ in an order O (on $\mathcal{G}$) for hybrid serializability, denoted $GEG_h(S_{\mathcal{G}}^O)$, can be defined by combining the conditions set forth in Definition 3 and Definition 7. A similar global execution theorem can also be derived assuming that the set $\mathcal{G}$ of global transactions is hybrid in an order. Rather than reiterating these formulations, we provide the following illustrative example:

Example 5 *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:*

$G_1 : w_{g_1}(a) r_{g_1}(b)$,

$G_2 : r_{g_2}(a) w_{g_2}(c) r_{g_2}(b)$,

$G_3 : r_{g_3}(a) r_{g_3}(c) r_{g_3}(b)$,

$G_4 : w_{g_4}(a) r_{g_4}(c)$,

which is hybrid in the order $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4$, where at local site LS_1, $G_{11} \overset{c}{\sim} G_{21} \subseteq G_{31} \overset{c}{\sim} G_{41}$ and at local site LS_2, $G_{12} \subseteq G_{22} \overset{c}{\sim} G_{32} \supseteq G_{42}$. Let L_{21} be a local transaction submitted at local site LS_2:

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

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : w_{g_1}(a) r_{g_2}(a) r_{g_3}(a) w_{g_4}(a)$,

$S_2 : w_{L_{21}}(b) r_{g_1}(b) r_{L_{21}}(c) w_{g_2}(c) r_{g_3}(c) r_{g_3}(b) r_{g_4}(c) r_{g_2}(b)$.

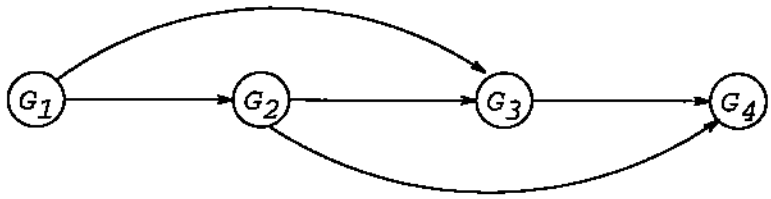
The global subschedule $S_{\mathcal{G}}$ is hybrid serializable in the order $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4$, and S is serializable. Note that, if the execution order of key operations which determine the hybrid relationships among global transactions are maintained:

$$w_{g_1}(a) \prec_{co}^{S_1} r_{g_2}(a) \prec_{co}^{S_1} r_{g_3}(a) \prec_{co}^{S_1} w_{g_4}(a)$$

$$r_{g_1}(b) \prec_{co}^{S_2} r_{g_2}(b)$$

$$w_{g_2}(c) \prec_{co}^{S_2} r_{g_3}(c) \prec_{co}^{S_2} r_{g_4}(c)$$

then global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems. In $GEG_h(S_g^O)$, *we have:*



□

In summary, hybrid serializability can be maintained by holding the execution order of hybrid operations of global transactions consistent with the order of their hybrid property. Thus, global concurrency control is actually simplified. With this basis, it is only necessary to enforce the hybrid property on global transactions, an issue which will be addressed in Section 5.3.

5.2 Optimality

Part of the attractiveness of hybrid serializability stems from our interest in defining all possible globally serializable schedules that can be determined without violation of local autonomy. Its efficacy in achieving this result is illustrated in the following discussion.

A property P of global transactions is defined as *optimal*[6] if there is no other property that is strictly weaker than P. We say that a property P_1 is weaker than a property P_2 if a set of global transactions that satisfies P_2 also satisfies P_1; that is, if P_2 implies P_1. A property P_1 is strictly weaker than a property P_2 if P_1 is weaker than P_2 and if P_2 is not weaker than P_1.

We claim that the application of the hybrid property H of global transactions to global transaction scheduling provides an optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site. That is, no other property is strictly weaker than H and allows the GDBS to indirectly determine the serialization order of global subtransactions at a local site[7]. This is formally proven in the following theorem:

[6] A similar definition has been suggested in [Wei89].

[7] Note that, if P is strictly weaker than H, then there exists a set of global transactions that satisfies P and does not satisfy H.

Theorem 7 (Optimal condition) *For systems using the methods described in this paper, the hybrid property of global transactions is an optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing any restrictions on or requiring any information from local sites other than local serializability.*

Proof: Let local concurrency controllers generate only locally serializable schedules. The proof proceeds by contradiction. Suppose the hybrid property H of global transactions is not optimal. There then exists a property P of global transactions that is strictly weaker than the hybrid property, and the serialization order of global subtransactions at a local site is determined at the global level by controlling the execution of the global transactions. A generic counter-example shows, however, that such a property does not exist.

Suppose that, at a local site LS_k, a set $\mathcal{G}_k$ of global subtransactions satisfies P and does not satisfy H. Let G_{1k} and G_{2k} be two global subtransactions in $\mathcal{G}_k$ that are not hybrid in any order. We have $G_{1k} \not\rightarrow G_{2k}$, $D_{G_{1k}} \not\subseteq D_{G_{2k}}$ and $D_{G_{1k}} \not\supseteq D_{G_{2k}}$. Then, there exist two different data items x and y, such that G_{1k} accesses x and does not access y, and G_{2k} accesses y and does not access x. We construct a local transaction $L_{k1} : w(x)w(y)$. The local concurrency controller at LS_k may produce the following locally serializable schedule:

$$S_k \ : \ w(x)G_{1k}G_{2k}w(y)$$

Note that S_k is conflict equivalent to $G_{2k}w(x)w(y)G_{1k}$ since G_{1k} does not conflict with G_{2k} and $w(y)$ and G_{2k} does not conflict with G_{1k} and $w(x)$. Then, S_k is serializable in the order $G_{2k} \rightarrow L_{k1} \rightarrow G_{1k}$. On the other hand, the local concurrency controller may produce the following locally serializable schedule:

$$S_k \ : \ G_{1k}w(x)w(y)G_{2k}$$

Then, S_k is serializable in the order $G_{1k} \rightarrow L_{k1} \rightarrow G_{2k}$. Consequently, the serialization order of the global subtransactions responds dynamically to the interactions of the local transaction, even though the execution order of global subtransactions remains consistent in both cases. Hence, the hybrid property provides an optimal condition for the determination of the serialization order of global subtransactions at a local site without imposing any restrictions on or requiring any information from local sites other than local serializability.

The generality of the above counter-example also implies that, for any set of global transactions which is not hybrid, the serialization order of its subtransactions at a local site may not be

determined at the global level. Hence, the hybrid property is also the only weakest property with which we are concerned. □

Therefore, no other property of global transactions can be strictly weaker than the hybrid property and can be applied as a sufficient condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing any restrictions on or requiring any information from local sites.

As thus defined through the above novel feature of the hybrid property, hybrid serializability, therefore encompasses the maximum set of globally serializable schedules that can be determined without violation of local autonomy. Hybrid serializability articulates the theoretical feasibility of global serializability in the MDBS environment. The optimal condition presented above implies that hybrid serializability defines the upper bound on the set of globally serializable schedules that can be generated without violation of local autonomy. However, an efficient protocol to implement this approach still awaits development.

5.3 Forcing the Hybrid Property in Global Transactions

As pointed out earlier, the chain-conflicting and fully sharing properties present the drawback of appending unnecessary updating operations or exponentially increasing appendices of extra retrieval operations. By combining the best features of these two properties, the hybrid property not only presents an optimal formulation but also offers a novel approach to compensating for the weakness of both previous methods. This is illustrated as follows:

Suppose we enforce the hybrid property on general global transactions by a particular order[8]. We append extra retrieval operations only if no hybrid order can be found between two global subtransactions. These appendices may render a subtransaction unwieldy, but they also increase the likelihood that it will conflict with or to be fully sharing with ($\supseteq$) the following subtransaction. Therefore, no extra operations may need be appended to the following subtransaction. The problem of exponentially increasing appendices is thus automatically avoided. The following example details these concepts.

Example 6 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b, c, d are in D_2. The following globally non-hybrid global transactions are submitted to the*

[8]This may be either first-come-first-serve, which enforces a hybrid order identical to the submitting order, or best-fit, which groups the global transactions and determines the most efficient hybrid order.

GDBS in the order G_1, G_2, G_3, G_4, G_5*:*

$G_1 : w_{g_1}(a) r_{g_1}(b)$

$G_2 : r_{g_2}(a) r_{g_2}(c)$

$G_3 : r_{g_3}(a) r_{g_3}(d)$

$G_4 : w_{g_4}(a) r_{g_4}(b)$

$G_5 : r_{g_5}(a) w_{g_5}(b)$

After appending extra retrieval operations in first-come-first-serve order, we get:

$$\left.\begin{array}{l} G_1 : w_{g_1}(a) r_{g_1}(b), \\ G_2 : r_{g_2}(a) r_{g_2}(c) \underbrace{r_{g_2}(b)}_{appended}, \\ G_3 : r_{g_3}(a) r_{g_3}(d) \underbrace{r_{g_3}(b) r_{g_3}(c)}_{appended}, \end{array}\right\} \textit{increasing appendices}$$

$$\left.\begin{array}{l} G_4 : w_{g_4}(a) r_{g_4}(b), \\ G_5 : r_{g_5}(a) w_{g_5}(b), \end{array}\right\} \textit{reducing appendices}$$

which is hybrid in the order $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4 \rightarrow G_5$*, where, at site* LS_1*,* $G_{11} \overset{c}{\sim} G_{21} \subseteq (or \supseteq) G_{31} \overset{c}{\sim} G_{41} \overset{c}{\sim} G_{51}$*, and at site* LS_2*,* $G_{12} \subseteq G_{22} \subseteq G_{32} \supseteq G_{42} \overset{c}{\sim} G_{52}$*.* □

Typically, the phases involving increasing and reducing appendices alternate, thus avoiding the spectre of exponentially increasing appendices. Furthermore, no extra updating operation needs to be appended to global transactions.

The extra operation method is presented here only as a theoretical vehicle to illustrate the possibility of generalization of the hybrid property to all global transactions. A detailed analysis of the enforcement of the hybrid property on global transactions is eschewed in lieu of a formal treatment of global concurrency control. As mentioned earlier, it is possible to enforce the hybrid property in a manner which appears completely transparent to application programmers. Details regarding the enforcement of the hybrid property of global transactions and the maintenance of the hybrid serializability of global subschedules will appear elsewhere.

6 Related Issues

In this section, other issues of interest will be discussed; in particular, the relationship of hybrid serializability to other suggested approaches and its adaptability to failure-prone multidatabase

environments.

6.1 Relationship to Other Work

Many approaches have been proposed to the problem of global concurrency control in MDBSs. Among these, two-level serializability [MRKS91] and quasi-serializability [DE89] characterize two correctness criteria for global schedules which maintain global consistency without imposing any restrictions on local sites. In this section, we compare the present work with these two correctness criteria.

Both two-level serializability and quasi-serializability relax global serializability to a certain degree. Informally, a global schedule S is two-level serializable if S restricted to each local site is serializable and S restricted to the set of global transactions in S is also serializable. A global schedule is quasi-serial if and only if it is serializable when restricted to each local site and there is a total order of its global transactions such that, if T_i precedes T_j in the order, all operations of T_i are executed before those of T_j in each local schedule; and a global schedule S is quasi-serializable if it is equivalent to a quasi-serial schedule of the same set of transactions. Although both criteria simplify the problem of global concurrency control, they can only maintain some degree of global consistency in certain restricted applications.

Let $\mathcal{H}$ denote the set of all possible global schedules; 2LSR denotes the set of two-level serializable global schedules; QSR denotes the set of quasi-serializable global schedules; SR denotes the set of serializable global schedules; CCSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are chain-conflicting serializable; SHSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are sharing serializable; and HSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are hybrid serializable.

As stated in [MRKS91] and [DE89], 2LSR is a superset of QSR, and QSR is a superset of SR. As pointed out earlier in this paper, HSR is a subset of SR and a superset of both CCSR and SHSR. There is no inclusive relationship between CCSR and SHSR. Note that the set of global schedules generated by the Optimistic Ticket Method (OTM) [GRS91] is a subset of CCSR. Figure 2 depicts the relationships among these different types of global schedules.

If the set of all global transactions submitted at the global level is chain-conflicting, the problem of global transaction scheduling is further reduced to maintaining the serializability of global transactions in a certain order. This is a sufficient condition for two-level serializability, which then

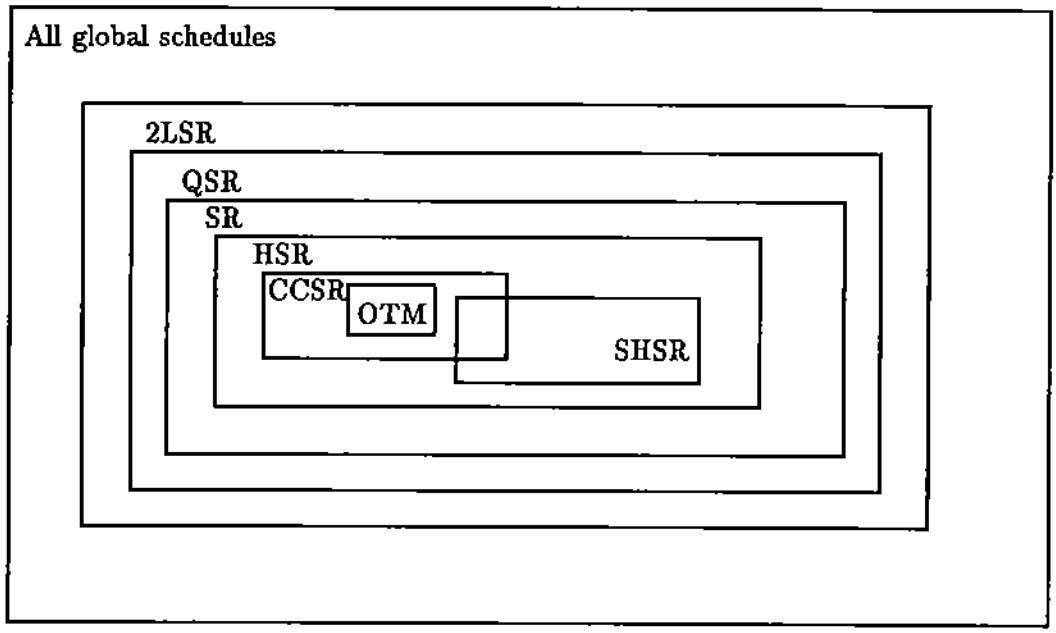


Figure 2: The relationships among 2LSR, QSR, SR, HSR, CCSR, SHSR, OTM

maintains global serializability. Thus, enforcing hybrid property on global transactions simplifies the problem of global concurrency control and global serializability is still retained.

6.2 Effects of Failures on Global Concurrency Control

The proposed criteria for global schedules have been defined in a manner appropriate to an environment involving transaction and other failures. That is, only committed global transactions are considered. Since any uncommitted global transaction may abort or a system failure may cause such transactions to abort at local sites, resubmission of aborted global transactions may result in an execution order of global transactions which is different from their original execution order. An irremedially non-serializable schedule may therefore be produced. The following example illustrates this situation.

Example 7 Consider an $MDBS$ consisting of two $LDBSs$ on D_1 and D_2 where data item a is in D_1 at LS_1, and c is in D_2 at LS_2. The following global transactions G_1 and G_2 are submitted:

$G_1 : r_{g_1}(a) r_{g_1}(c)$

$G_2 : w_{g_2}(a) w_{g_2}(c)$

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : r_{g_1}(a)\ w_{g_2}(a)\ c_{g_{21}}\ \underbrace{*****}_{failure}$

$S_2 : r_{g_1}(c)\ w_{g_2}(c)\ c_{g_{12}}\ c_{g_{22}}$

That is, for some reason global subtransaction $G_{11} : r_{g_1}(a)$ is aborted before it commits. It cannot therefore be re-executed without rendering the global schedule S non-serializable. Note that, in this case, there is even no local transactions to be considered in S. □

Thus, a protocol for hybrid serializability in a failure-prone multidatabase environment must take into account the effects of failures and be able to recover from such effects. It follows from this that the commit order of global subtransactions must be consistent with their serialization order. A uniform theory of global concurrency control and failure recovery ensues. Moreover, this theory must be compatible with the preservation of atomicity of global transactions. A detailed analysis of global concurrency control in failure-prone environments and of atomic commitment can be found in [EJK91, ZJ93].

7 Conclusions

To date, there has been no theoretical study of the maintenance of global serializability through global transaction scheduling in the MDBS environment. Existing approaches to global concurrency control in MDBSs either relax the serializability theory or impose restrictions on local concurrency control mechanisms. In this paper, we have proposed three global transaction scheduling criteria to maintain global serializability without imposing any additional restrictions on LDBSs other than local serializability. These three criteria are chain-conflicting serializability, sharing serializability, and hybrid serializability.

We have therefore:

- Formally proposed and proved a theory of global concurrency control for maintaining global serializability in multidatabase systems without placing any additional restrictions on local sites other than local serializability; and
- Indicated the upper limit on global serializability while maintaining local autonomy.

As an outgrowth of these criteria, we have shown that global serializability can be ensured at the global level by utilizing the intrinsic characteristics of global transactions. The mixed structral

features of the hybrid property of global transactions determines a sufficient condition for the GDBS to synchronize the serialization orders of global transactions at all local sites without violation of local autonomy. Moreover, global concurrency control is simplified by controlling the execution order of the hybrid operations that determine the hybrid property of global transactions. We have also shown that global concurrency may be limited if local autonomy is a major factor to be considered in MDBSs.

Thus, the hybrid property of global transactions is considered to be the fundamental structural feature of global transactions necessary for achieving global serializability without violation of local autonomy. The central issue of global concurrency control therefore becomes the enforcement of the hybrid property on global transactions. A ticket method [GRS91] is proposed to force conflicts among all global transactions, thus generating a strong implementation of the hybrid property. An extra operation method is also proposed in this paper to enforce the hybrid property on global transactions. The extra operation method enforces the hybrid property on global transactions in a manner transparent to application programmers. Protocols are currently being developed to implement this method.

In order to implement hybrid serializability in a failure-prone multidatabase environment, the commit order of global subtransactions must obey their serialization order. Moreover, preservation of the atomicity of global transactions may be ensured through atomic commitment protocols. The results of these investidations are presented elsewhere.

In summary, hybrid serializability formulates the maximum set of globally serializable schedules to be determined in MDBSs without violation of local autonomy, thus clarifying the potential limits of the global concurrency controller within the constraints of local autonomy.

Acknowledgements

This paper is based in part on ([ZE93]). The authors have benefitted greatly from discussions in the InterBase group at Purdue University. We especially thank Xiangning Liu for taking the time to read this paper and for pointing out some minor errors in an earlier version of this work.

References

[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), 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.

[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.

[EJK91] A. Elmagarmid, J. Jing, and W. Kim. Global committability in multidatabase systems, 1991. (submitted for publication, also available as Purdue University Technical Report CSD-TR-91-017).

[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.

[GPZ86] V. Gligor and R. Popescu-Zeletin. Transaction management in distributed heterogeneous database management systems. *Information Systems*, 11(4):287–297, 1986.

[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.

[MRB+92] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. The concurrency control problem in multidatabases: Characteristics and solutions. Technical Report TR-91-37, University of Texas at Austin Department of Computer Science, 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.

[Pap86] C. Papadimitriou. *The Theory of Database Concurrency Control.* Computer Science Press, 1986.

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

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

[Wei89] William Weihl. Local atomicity properties: Modular concurrency control for abstract data types. *ACM Transactions on Programming Languages and Systems*, 11(2):249–282, April 1989.

[ZE93] 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*, San Diego, California, January 1993. (to appear, also available as Purdue university Technical Report CSD-TR-92-039).

[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, also available as Purdue university Technical Report CSD-TR-92-042).

Appendix A

The proof of Theorem 1:

(*if*) Assume that there exists a total order O on global transactions in S, and for every local site $LS_k(1 \leq k \leq m)$, the serialization order of global subtransactions in S_k is consistent with O. We construct the serialization graph SG for S, denoted $SG(S)$, as a directed graph whose nodes are the transactions in S and whose edges are all $T_i \rightarrow T_j (i \neq j)$ on both global and local transactions, such that one of T_i's operations precedes and conflicts with one of T_j's operations in S. We need to prove that $SG(S)$ is acyclic [BHG87].

Suppose there is a cycle in $SG(S)$. Without loss of generality, let the cycle be $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_k \rightarrow T_1 (k > 1)$. These edges imply that in S, T_1 appears before T_2, which appears before T_3, which appears $\cdots$ before T_k, which appears before T_1. Since each local subschedule of S is serializable and there is no conflict between local transactions at one site and local transactions (or global subtransactions) at another site, there must be a set of global transactions $\{G_{i_1}, G_{i_2}, \cdots, G_{i_r}\} \subseteq \{T_1, T_2, \cdots, T_k\}$ such that G_{i_1} precedes G_{i_2}, G_{i_2} precedes G_{i_3}, $\cdots$, G_{i_r} precedes G_{i_1}. There is therefore no total order on global transactions such that G_{i_1} precedes G_{i_2}, G_{i_2} precedes G_{i_3}, $\cdots$, G_{i_r} precedes G_{i_1} at the same time. This is contradictory to our assumption. Hence, $SG(S)$ is acyclic. By the serialization theorem given in [BHG87], S is serializable.

(*only if*) Assume that S is serializable in a total order O. Then, for each local site LS_k $(1 \leq k \leq m)$, the serialization order of S_k is consistent with O. Let O' be O restricted to the global transactions in S. Consequently, the serialization order of global subtransactions at each local site LS_k $(1 \leq k \leq m)$ is consistent with O'. Hence, we prove the theorem. □