1993

# Global Committability in Multidatabase Systems

O. Bukhres

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

J. Jing

W. Kim

A. Zhang

Report Number:
93-052

# GLOBAL COMMITABILITY IN MULTIDATABASE SYSTEMS

O. Bukhres
A. Elmagarmid
J. Jing
W. Kim
A. Zhang

CSD-TR-93-052
August 1993

# Global Committability in Multidatabase Systems

O. Bukhres, A. Elmagarmid, J. Jing*, W. Kim†, and A. Zhang
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 USA

**Abstract**

The preservation of local autonomy and the atomic commitment of global transactions present conflicting exigencies to the design of multidatabase transaction management systems. In this paper, we investigate a forward recovery approach to the atomic commitment of global transactions while at the same time preserving local autonomy. A theoretical basis for the application of forward recovery to the atomic commitment of global transactions is developed. In particular, we examine the effect on atomic commitment of the intrinsic semantics of global transactions, as manifested in value dependency relationships. The atomicity of a global transaction is thus ensured through a controlled commitment order of its global subtransactions, while its aborted global subtransactions are retried. A global atomic commitment protocol is grounded upon the proposed theory.

**Index terms:** transaction management, multidatabase, global committability, compensation, global and local transactions.

## 1 Introduction

The preservation of the atomicity or semantic atomicity [7] of global transactions in a multidatabase system (MDBS) has been recognized as a substantial and as yet unresolved challenge [15, 8]. A global transaction in this context consists of a series of global subtransactions, with each global subtransaction being executed at a component database system of the MDBS. The goal of atomic commitment in this context is to ensure that either all of the effects of each global transaction are made permanent in databases or any partial effect of a global transaction is undone to retain multidatabase consistency. A two-phase commit (2PC) protocol has been proposed for traditional distributed database systems to ensure the atomicity of global transactions [1]. 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. It has been shown in [10, 14] that the 2PC protocol is inadequate to the maintenance of the atomicity of global transactions in the MDBS environment. Some local database systems may not support a prepare-to-commit state. It

*Intergraph Corporation, Huntsville, AL 35894
†UniSQL, Inc., 9390 Research Blvd., Austin, TX 78759

may also be a violation of local autonomy to require local database systems to provide prepare-to-commit states. Thus, the difficulty of ensuring that a single logical action (commit or abort) of a global transaction is consistently carried out at multiple local sites is considerably increased by the demands of local autonomy.

In this paper, we investigate a theoretical basis for the atomic commitment of global transactions in the MDBS environment in which the local database systems are required only to ensure serializability and recoverability [1]. In the proposed formulation, the atomicity of global transactions is ensured through an extension of the retry approach [13]. This methodology differs from the retry approach proposed in [13], where the execution of a global transaction at one site is independent of its execution at other sites, with no value dependencies [5] present among the subtransactions of a global transaction. In contrast, the present theory permits value dependencies to be defined among the subtransactions of a global transaction. This investigation of the effect of the value dependencies of global transactions on global transaction management is strongly motivated by the nature of applications. Commonly, many applications involve data transfer among different local database sites, which will result in value dependencies among the subtransactions of a global transaction.

The formulation of this theory rests upon the observation that the semantic information inherent within global transactions may be incorporated into the retry approach as a means to control the commitment order of the global subtransactions of each global transaction. A fundamental property of global histories, termed *global committability*, is formulated which defines a necessary condition for a global subtransaction to be retried without violation of multidatabase consistency. The class of global transactions that can be executed in the MDBS environment is thereby extended by allowing value dependencies to be defined on global transactions. An atomic commitment protocol based upon this theory is proposed which ensures the atomicity of global transactions in the MDBS environment.

This paper is organized as follows. Section 2 introduces the system model and the terminology to be employed, while Section 3 discusses the crucial problems presented by the retry technique. In Section 4, we propound a theory of global committability. A protocol that implements the proposed theory, enabling reliable global transaction management, is outlined in Section 5. A discussion and concluding remarks are offered in Sections 6 and 7.

## 2 The Multidatabase System and Related Terminology

We shall consider an MDBS to comprise of a set of $\{\text{LDBS}_i, \text{ for } 1 \leq i \leq m\}$, where each $\text{LDBS}_i$ is a pre-existing autonomous database management system on a set of data items at 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. As a necessary assumption of this paper, we presume that the concurrency control and recovery mechanisms of LDBSs ensure serializability and recoverability [1]. However, no restriction is imposed on these mechanisms.

We assume that the GTM submits global transaction operations to the LDBSs through the servers, which therefore 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 by controlling their submission.

Following [1], we assume the availability of four basic database 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. We shall alternatively use $r(x, v)$ (or $w(x, v)$) to denote an operation which reads (or writes) a value $v$ from (or to) data item $x$. 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 that is the maximum (last) element in the partial order.

In the MDBS environment, a local transaction is a transaction that accesses the data items at a single local site. A global transaction is a set of global subtransactions, within which each global subtransaction is a transaction that contains all operations accessing the data items at a single local site. A global transaction may contain more than one termination operation, with one such operation provided for each subtransaction. A global subtransaction $G_{ij}$ denotes a global subtransaction of $G_i$ accessing LDBS$_j$. We say that $G_{ij_t}$ is *value dependent* on $G_{ij_1}, ..., G_{ij_{t-1}}$ $(1 \leq j_1, ..., j_t \leq m)$, denoted $G_{ij_1} \rightarrow_{vd} G_{ij_t}$, $G_{ij_2} \rightarrow_{vd} G_{ij_t}$, ..., $G_{ij_{t-1}} \rightarrow_{vd} G_{ij_t}$, if the execution of one or more operations in $G_{ij_t}$ is determined by the values read by $G_{ij_1}, ..., G_{ij_{t-1}}$. We assume that value dependencies are the only relationships defined among the global subtransactions of each global transaction.

A history over a set of transactions is a partial order of all and only the operations of those transactions which orders all conflicting operations and respects the order of operations specified by the transactions. A more formal definition of a history can be found in [1]. A local history $H_k$ is a history over both local transactions and global subtransactions which are executed at local site $LS_k$. A global history $H$ is a history over both local and global transactions which are executed in an MDBS. $C(H)$ denotes $H$ restricted to the committed transactions in $H$. A global subhistory $H_{\mathcal{G}}$ is $H$ restricted to the set $\mathcal{G}$ of global transactions in $H$. We denote $o_1 <_H o_2$ if operation $o_1$ is executed before operation $o_2$ in history $H$.

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* if it preserves these database integrity constraints. A multidatabase state is *consistent* if it preserves all integrity constraints defined in the MDBS environment.

# 3 Forward Recovery Approach

A forward recovery approach which utilize the redo and retry techniques has been proposed in the literature to address the issue of atomic commitment in MDBSs. The redo technique initially proposed in [4] and later elaborated in [2, 12] acts as a pseudo-2PC, with servers rather than the LDBSs considered as the participants. If a global subtransaction is aborted by an LDBS after the GTM has decided to commit the global transaction, the server at this local site submits a *redo transaction* to the LDBS for execution. This redo transaction consists of all the write operations performed by the global subtransaction. Multidatabase

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 technique requires that the data items accessed by global subtransactions must be different from the data items accessed by local transactions at a local site. In the present context, since we are considering the environment in which the updating of data items is not clearly differentiated with respect to global and local transactions, we shall therefore focus our investigations upon the application of the retry technique.

The retry technique as applied to the preservation of the atomicity of global transactions allows each global subtransaction to commit unilaterally and requires the retrial of aborted global subtransactions. We say that a global subtransaction is *retriable* if it is guaranteed to commit after a finite number of retrials when executed from any consistent database state. This retriability does not guarantee that the commitment of a retried global subtransaction will always ensure multidatabase consistency. Some difficulties may arise if one global subtransaction has a value dependency relationship with another global subtransaction. If local autonomy prevents the global transaction manager from blocking the execution of local transactions after a global subtransaction aborts but before it is retried, then the execution of such local transactions may result in the resubmission of the subtransaction creating multidatabase inconsistencies. The following example illustrates this situation.

**Example 1** *Consider an MDBS that has data item $a$ in $LS_1$ and $b$ in $LS_2$. Let the integrity constraint be $a + b = total$ for a data item total. Suppose a transferring global transaction below is executed:*

$G_1 : r_{G_{12}}(b) r_{G_{11}}(a) w_{G_{12}}(b, a + b) w_{G_{11}}(a, 0)$

*where $G_{11} \multimap_{vd} G_{12}$. Let us consider a scenario in which $G_{12}$ commits and $G_{11}$ aborts. If a local transaction at $LS_1$ is executed to update data item $a$ before $G_{11}$ is retried, then the retrial of $G_{11}$ at $LS_1$ may read a value of data item $a$ that is different from the value of data item $a$ read by the original execution of $G_{11}$. Thus, the execution of the retried $G_{11}$ may result in multidatabase inconsistency.* □

To prevent such inconsistencies, an approach is proposed in [13] that stipulates that no value dependencies may exist between the subtransactions of a global transaction. In this formulation, the execution of a global transaction at one local site is semantically independent of its execution at other local sites.

Another anomaly of the retry technique appears when serializability is held to be the correctness criterion for the execution of local and global transactions.

**Example 2** *Consider an MDBS that has data item $a$ in $LS_1$ and data item $b$ in $LS_2$. Let global transactions $G_1$ and $G_2$ be submitted:*

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

*The following global history is then serializable:*

$H : w_{G_{11}}(a) w_{G_{21}}(a) w_{G_{12}}(b) w_{G_{22}}(b).$

*Suppose that $G_{11}$ and $G_{22}$ successfully commit, but $G_{12}$ and $G_{21}$ are aborted before $c_{G_{12}}$ and $c_{G_{21}}$ are executed due to failures at local sites $LS_1$ and $LS_2$. At this point, the global history becomes:*

$$H' : w_{G_{11}}(a) w_{G_{22}}(b) c_{G_{11}} c_{G_{22}} \underbrace{* * * * *}_{LS_1, LS_2 fail} .$$

*The subtransactions $G_{12}$ and $G_{21}$ cannot be re-executed without causing the execution of global transactions $G_1$ and $G_2$ to be non-serializable. This results in a situation in which $G_{11}$ and $G_{22}$ have committed but the retrial of $G_{12}$ and $G_{21}$ in any order will result in a non-serializable global history.* □

Problems such as those illustrated above arise in the MDBS environment as a result of the requirements of local autonomy. Globally uncontrolled local transactions at each local site may be executed in an interleaved fashion with global subtransactions, and local sites may unilaterally abort global subtransactions without agreement from the GTM. We must therefore develop a method which permits the GTM to guarantee the retriability of global subtransactions while preserving multidatabase consistency and local autonomy.

# 4 Global Committability

In this section, we shall investigate a method of forward recovery, which is an extension of the retry technique to atomic commitment. This method of forward recovery defines a property on global histories, termed global committability, which preserves the atomicity of global transactions by scheduling the commitment order of global subtransactions and retrying (or resubmitting) aborted global subtransactions. Global committability facilitates the definition of value dependencies in global transactions. No restrictions other than serializability and recoverability need be placed on local sites.

Our discussion is predicated upon the assumption that serializability [1] is maintained as the correctness criterion for the execution of local and global transactions. Let $T_1 \prec_{sr}^{H} T_2$ denote that transaction $T_1$ precedes transaction $T_2$ in the serialization order of history $H$. Following [9, 11, 16], a global history $H$ is serializable if and only if the serialization orders of global subtransactions at all local sites $LS_k$ for $1 \leq k \leq m$ are relatively synchronized[1]. Global committability is developed based upon this necessary and sufficient condition.

Let $H$ be a global history and $H_{\mathcal{G}}$ be $H$ restricted to the set $\mathcal{G}$ of global transactions in $H$. Following the system model proposed in Section 2, the GTM can control the scheduling of $H_{\mathcal{G}}$ by controlling the submissions of operations of global transactions. We have seen that, at the global level, an aborted global subtransaction may not be retriable without violating multidatabase consistency, due to the effect of the interleaving of local transactions at local sites. To ensure the retriability of each global subtransaction without both violating multidatabase consistency and placing restrictions at local sites, we suggest that the commitment order of global subtransactions at the global level be scheduled so that the interleaving of globally uncontrolled local transactions would not affect the retrial of global subtransactions.

In this context, we need only consider the effect of the interleaving of those local transactions which are appended to the committed projection of any prefix $H'_{\mathcal{G}}$ of a global subhistory $H_{\mathcal{G}}$. When an aborted global subtransaction is retried, any uncommitted global subtransactions in $H'_{\mathcal{G}}$ can be aborted at the global level and resubmitted for execution. We therefore must ensure that every uncommitted global subtransaction remains retriable without violating multidatabase consistency after arbitrary local transactions are appended to $C(H'_{\mathcal{G}})$ of any prefix $H'_{\mathcal{G}}$ of a global subhistory $H_{\mathcal{G}}$. We say that a global subhistory $H_{\mathcal{G}}$ is *prefix local extension-closed* if, when any $C(H'_{\mathcal{G}})$, where $H'_{\mathcal{G}}$ is a prefix of $H_{\mathcal{G}}$, is interpenetrated by the operations of local

[1] That is, if $G_{ij}, G_{il}, G_{kj}$ and $G_{kl}$ exist, then $G_{ij} \prec_{sr}^{H_j} G_{kj}$ if and only if $G_{il} \prec_{sr}^{H_l} G_{kl}$.

transactions that follow the criteria for the execution of transactions at local sites, every uncommitted global subtransaction in $H'_{\mathcal{G}}$ can be retried without violating multidatabase consistency. Following this concept, preserving the prefix local extension-closed property of global subhistories ensures that each uncommitted global subtransaction can be retried without violating multidatabase consistency, even when arbitrary local transactions are inserted into the execution of global transactions.

We shall now develop the definition of global committability. Two aspects of global committability may first be established with respect to a single global transaction and a set of global transactions:

**Definition 1 (Intra-committability)** *A global subhistory $H_{\mathcal{G}}$ is intra-committable if, for every global transaction $G_i$ in $\mathcal{G}$ and any two global subtransactions $G_{ij}$ and $G_{ik}$ of $G_i$, $G_{ij} \rightarrow_{vd} G_{ik}$ and $c_{ik} \in H_{\mathcal{G}}$ imply $c_{ij} <_{H_{\mathcal{G}}} c_{ik}$.* □

In other words, an intra-committable global subhistory requires that the commitment order of the global subtransactions of a global transaction be consistent with the dependency order of value dependency relationships between them.

**Definition 2 (Inter-committability)** *A global subhistory $H_{\mathcal{G}}$ is inter-committable if, for any two global subtransactions $G_{ik}$ and $G_{jk}$ of different global transactions in $\mathcal{G}$ at local site $LS_k$, $G_{ik} \prec_{sr}^{H_k} G_{jk}$ and $c_{jk} \in H_{\mathcal{G}}$ imply $c_{ik} <_{H_{\mathcal{G}}} c_{jk}$.* □

In other words, an inter-committable global subhistory requires that the commitment order of global subtransactions of different global transactions be consistent with their serialization order at a local site.

Global committability of global histories arises as a combination of intra-committability and inter-committability:

**Definition 3 (Global committability)** *A global subhistory $H_{\mathcal{G}}$ is globally committable if it is both intra-committable and inter-committable.* □

Following the above three definitions, if global subhistory $H_{\mathcal{G}}$ is globally committable, then each global subtransaction in $H_{\mathcal{G}}$ can only commit after all global subtransactions upon which it is value dependent and all global subtransactions which precede it in the serialization order at its local site have committed. We claim that, if global subhistories are globally committable, then they are prefix local extension-closed. Consequently, every uncommitted global subtransaction is retriable at the global level without both violating multidatabase consistency and placing restrictions on local sites. The following theorem is illustrative:

**Theorem 1** *If a global subhistory $H_{\mathcal{G}}$ is globally committable, then $H_{\mathcal{G}}$ is prefix local extension-closed.*

**Proof:** Let $H_{\mathcal{G}}$ be a global subhistory and $C(H'_{\mathcal{G}})$ be the committed projection of any prefix $H'_{\mathcal{G}}$ of $H_{\mathcal{G}}$. Without loss of generality, suppose a global subtransaction $G_{ij}$ in $\mathcal{G}$ is executed but not committed in $H'_{\mathcal{G}}$. We need to show that $G_{ij}$ can be retried without violating multidatabase consistency. We first consider the effect of the retrial of $G_{ij}$ on the execution of $G_i$ in $H'_{\mathcal{G}}$. Since $H_{\mathcal{G}}$ is globally committable, any global

subtransactions of $G_i$ which are value dependent on $G_{ij}$ must not have committed in $H'_{\mathcal{G}}$. Let $G_{il}$, which is value dependent on $G_{ij}$, also be executed in $H'_{\mathcal{G}}$ but not yet be committed. $G_{il}$ can be aborted and resubmitted for execution if the retrial of $G_{ij}$ would result in inconsistency in the execution of $G_{il}$. As our model assumes that value dependencies are the only relationships in effect among the global subtransactions of each global transaction, $G_{ij}$ can be retried without violating multidatabase consistency relative to other global subtransactions of $G_i$. We then consider the effect of the retrial of $G_{ij}$ on the execution of global transactions other than $G_i$. Since $H_{\mathcal{G}}$ is globally committable, any global subtransactions of different global transactions from $G_i$ which must be serialized after $G_{ij}$ must not have committed in $H'_{\mathcal{G}}$. Let $G_{kj}$ be a global subtransaction which is executed in $H'_{\mathcal{G}}$ and serialized after $G_{ij}$ at local site $LS_j$. If the retrial of $G_{ij}$ results in a situation in which the serialization order of $G_{ij}$ will follow the serialization order of $G_{kj}$, then $G_{kj}$ can be aborted and resubmitted for execution. Thus, $G_{ij}$ can be retried without violating multidatabase consistency relative to other global subtransactions of different global transactions. Hence, $G_{ij}$ can be retried relative to all other global subtransactions without both violating multidatabase consistency and placing any restrictions on the local site. $H_{\mathcal{G}}$ is prefix local extension-closed. □

It is clear that, if all global subtransactions of global transactions are retriable without violating multidatabase consistency, then the atomicity of these global transaction is preservable. Thus, we have the following corollary:

**Corollary 1** *If a global subhistory $H_{\mathcal{G}}$ is globally committable and each global subtransaction in $\mathcal{G}$ is retriable, then the atomicity of global transactions in $\mathcal{G}$ can be preserved.*

The maintenance of the intra-committability of global subschedules at the global level is determined by the characteristics of the value dependencies defined on the global subtransactions of each global transaction. Such value dependencies can be described by a graph as follows:

**Definition 4 (Value dependency graph)** *A value dependency graph of global transaction $G_i$, denoted VDG($G_i$), is a directed graph whose nodes are all global subtransactions of $G_i$ and whose edges are all $G_{ij_1} \rightarrow G_{ij_2}$ ($G_{ij_1}, G_{ij_2} \in G_i$), such that $G_{ij_1} \rightarrow_{vd} G_{ij_2}$.* □

The acyclicity of value dependency graphs provides the necessary and sufficient conditions for maintaining global subhistories as intra-committable. More precisely, we have the following theorem:

**Theorem 2** *A global subhistory $H_{\mathcal{G}}$ is intra-committable if and only if $\forall G_i \in \mathcal{G}$, VDG($G_i$) is acyclic.*

**Proof:** (1) Assume that $\forall G_i \in \mathcal{G}$, VDG($G_i$) is acyclic. Then, for any $G_i$ in $\mathcal{G}$, VDG($G_i$) may be topologically sorted. Without loss of generality, let $G_{i1}, ..., G_{im}$ be the nodes of VDG($G_i$) and $j_1, ..., j_m$ be a permutation of 1,2,...,m such that $G_{ij_1}, G_{ij_2}, ..., G_{ij_m}$ is a topological sort of VDG($G_i$). This order ensures that the commitment orders of the global subtransactions of $G_i$ conform to the definition of intra-committability. To illustrate this, let $G_{il}$ and $G_{ik}$ be subtransactions of $G_i$ such that $G_{il}$ is value dependent upon $G_{ik}$. By the definition of VDG($G_i$), $G_{ik} \rightarrow G_{il}$ is an edge in VDG($G_i$). Thus, $G_{ik}$ must appear before $G_{il}$ in the topological sort $G_{ij_1}, G_{ij_2}, ..., G_{ij_m}$. If the commitment order of all subtransactions of $G_i$ follows the order of $G_{ij_1}, G_{ij_2}, ..., G_{ij_m}$, then $G_{ik}$ commits before $G_{il}$ commits. Hence, $H_{\mathcal{G}}$ is intra-committable.

(2) Assume that $H_{\mathcal{G}}$ is intra-committable. We need to prove that $\forall G_i \in \mathcal{G}$, VDG($G_i$) is acyclic. Suppose, for any $G_i$ in $\mathcal{G}$, there is a cycle in VDG($G_i$) and, without loss of generality, let that cycle be $G_{i1} \rightarrow G_{i2} \rightarrow , ..., \rightarrow G_{ik} \rightarrow G_{i1}$. These edges imply that $G_{i1}$ must commit before $G_{i2}$, which must commit before $G_{i3}$, which must commit ... before $G_{ik}$, which must commit before $G_{i1}$. As a result, the commitment of each global subtransaction rests upon the commitment of another subtransaction in the group of global subtransactions $G_{i1}, ..., G_{ik}$, producing a commitment waiting cycle. Thus, $G_i$ cannot be intra-committable, contradicting the initial assumption. Hence, $\forall G_i \in \mathcal{G}$, VDG($G_i$) must be acyclic. □

Theorem 2 precisely defines the limits on the application of the retry approach to global transactions which are allowed to have value dependencies among their subtransactions. In such a situation, full local autonomy is preserved, as no restriction need be placed upon local sites to maintain intra-committability.

We shall now turn our attention to the preservation of inter-committability on global subhistories. Here, the fundamental concern regards the possibility of determining the serialization orders of global subtransactions at the global level. Much research of both a theoretical and a practical nature has been directed to determining the serialization orders of global subtransactions [9, 3, 2, 16]. In particular, the ticket method [9] and the extra operation method [16] offer approaches to determining the serialization orders of global subtransactions at the global level while requiring only the maintenance of serializability at local sites. As these approaches are applicable to our scenario, further discussion of this subject will not be presented here. We assume that the serialization order of global subtransactions at each local site can be determined at the global level. The GTM can then ensure that the submission of commit operations of global subtransactions is consistent with their serialization order. Note that such control of the commitment order of global subtransactions will not conflict with the maintenance of recoverability at local sites. If there is a read from [1] relationship between two global subtransactions at a local site, then the serialization order of these global subtransactions is clearly consistent with their commitment order.

Thus, given that the value dependency graphs of global transactions are acyclic and that each global subtransaction commits after a sufficient number of retrials, every global transaction can commit. A commitment protocol can be designed to enforce global committability on global subhistories which preserves the atomicity of global transactions without violation of local autonomy. We shall discuss this issue in the next section.

# 5 Implementation Issues

Building upon the discussion in the previous section, we shall here assume that the value dependency graph of each global transaction is acyclic. A method of guaranteeing the acyclicity of the value dependency graph of each global transaction appears in [17, 6]. That research proposes a new transaction model for global transactions which permits each global transaction to have more than one subtransaction at a local site.

We here propose to enforce global committability through the global commit protocol. This protocol consists of two phases, a forward commit phase and a backward recovery phase. In order to implement the global commit protocol, the GTM must maintain a serialization order of global transactions and an acyclic value dependency order for each global transaction. The serialization order may be determined either

statically before the global transactions are executed or dynamically at run-time, based on the approach to be employed. Without loss of generality, we assume that the static serialization order is used, so that there is a pre-determined total serialization order as an input parameter for the global commit protocol.

The global commit protocol, as outlined below, is invoked after a global subtransaction has completed its read and write operations.

- **Forward Commit Phase:** When a global subtransaction $G_{ij}$ has completed its read and write operations at local site $LS_j$, the forward commit phase is initiated. This process determines: (1) whether any global subtransactions upon which $G_{ij}$ is value dependent have committed at local sites other than $LS_j$, and (2) whether any global subtransaction $G_{kj}$ which precedes $G_{ij}$ in the pre-determined serialization order has committed at local site $LS_j$. If both conditions (1) and (2) are satisfied, then the GTM submits the commit request of the global subtransaction to local site $LS_j$; otherwise, the commit operation of $G_{ij}$ is blocked until conditions (1) and (2) are satisfied.

- **Backward Recovery Phase:** A global subtransaction $G_{ij}$ at local site $LS_j$ may be aborted as a result of failures which have occurred at local sites. When such failures occur, the GTM will be informed of this abort and will then initiate the backward recovery phase. In this phase, abort commands are sent to the corresponding local sites for (1) any global subtransactions at local sites other than $LS_j$ that are value dependent upon the aborted subtransaction $G_{ij}$, and (2) any global subtransactions at local site $LS_j$ that follow $G_{ij}$ in the pre-determined serialization order.

  For any aborted global subtransaction, a recursive invocation of the backward recovery phase is executed such that conditions (1) and (2) above are satisfied. The forward commit phase serves to ensure that these global subtransactions have not been committed. At the close of the backward recovery phase, the GTM then resubmit the aborted subtransactions for execution.

The global commit protocol permits the GTM to submit global subtransactions to a local site in parallel. Of practical concern is that this protocol may lead to a cascading abort in the backward recovery phase; the abort of a global subtransaction may trigger the abort of further global subtransactions. For instance, the abort of a global subtransaction will lead to the abort of those global subtransactions which are value dependent upon or serialized after it. These latter aborted global subtransactions may similarly generate the aborting of further global subtransactions. If such a cascading abort will create serious difficulties in a particular application, a variation of the strict two-phase locking protocol should be used to control the submission of global transactions at the global level. Using this approach, if $G_{kj}$ precedes $G_{ij}$ in the serialization order, then those operations of $G_{ij}$ that conflict with those of $G_{kj}$ could only be submitted for execution after $G_{kj}$ has committed. A cascading abort would therefore be avoided among global subtransactions that belong to different global transactions. The reduced concurrency of this method would, however, lead to decreased efficiency. In each application, the respective drawbacks of some degree of cascading abort and low concurrency must be taken into consideration in the enforcement of the global commit protocol. For some applications, low concurrency may be preferable to a highly cascading abort; for others, a low degree of cascading abort may be chosen over high concurrency.

The global commit protocol provides an algorithm for maintaining the atomicity of global transactions without placing restrictions at local sites other than serializability and recoverability. One important feature of this protocol is the simplicity of its implementation. We thus expect atomic commitment to be efficiently implemented. In addition, the global commit protocol allows transactions to execute with less blocking than the 2PC protocol, especially when locking protocols [1] are employed at local sites. This is intuitively evident from the observation that global or local transactions can access the resources that have been used by a partially committed global transaction.

# 6 Discussions

Global committability ensures the atomicity of global transactions, provided that each global subtransaction is retriable. However, in practice, many global subtransactions are not retriable. The following example is illustrative:

**Example 3** *Consider an application typical of a travel agency. In this instance, a travel agent wishes to arrange a business trip for a customer, involving the reservation of one ticket through an airline database at one local site and of another ticket from a second airline database at a different local site. Suppose the two global subtransactions of the global transaction necessary to accomplish this task have made these reservations, but only one of them has actually committed. At this point, if the uncommitted global subtransaction aborts and a local transaction makes the same reservation, there may be no reserved space available for the retrial of the aborted global subtransaction. Consequently, this global subtransaction may result in an indefinite retrial. Thus, it is not retriable.* □

Forward recovery, however, may be extended by the incorporation of the backward recovery approach. For instance, the committed global subtransaction in Example 3 can be easily compensated by releasing the reserved ticket. A global subtransaction is compensatable if the effect of its execution at a local site can be removed by executing a compensating transaction. In [13], a unified method involving the retry and compensation approaches has been proposed. This method formulates each global transaction as the combination of a set of compensatable subtransactions, a set of retriable subtransactions, and a single pivot subtransaction which is neither compensatable nor retriable. Any of these three parts can be optional, and no value dependencies may exist among the global subtransactions of a global transaction. Following this global transaction model, the compensatable subtransactions must be committed before the commitment of the pivot subtransaction, which in turn must commit before the commitment of the retriable subtransactions. When the pivot subtransaction commits, the global transaction will commit; otherwise, the global transaction aborts and all committed compensatable subtransactions are compensated.

We can extend the above unified method by allowing value dependencies to be defined on global transactions and by combining global committability with compensation in global subhistories. Let $G_i^c$ denote the set of compensatable subtransactions in $G_i$, $G_i^r$ denote the set of retriable subtransactions in $G_i$, and $G_i^p$ denote the pivot subtransaction in $G_i$. Following the discussion of the previous section, the prohibition against the existence of value dependencies among retriable subtransactions can be relaxed by enforcing

global committability on the commitment order of retriable subtransactions in $G_i^r$, as long as VDG($G_i^r$) is acyclic. Furthermore, since the compensatable subtransactions must commit before the pivot subtransaction, which must in turn commit before the retriable subtransactions, each retriable subtransaction can be value dependent on the compensatable subtransactions or on the pivot subtransaction, the pivot subtransaction can be value dependent on the compensatable subtransactions, and each compensatable subtransaction can be value dependent on other compensatable subtransactions, as long as each compensatable subtransaction can still be compensated at a single local site. Note that both the compensatable subtransactions and the pivot subtransaction must not be value dependent on the retriable subtransactions, or a cyclic commitment dependency may be created among these global subtransactions. In such a dependency, a retriable subtransaction $G_{ij}$ must commit after the commitment of both the compensatable subtransactions and the pivot subtransaction, but $G_{ij}$ would also be required to commit before the compensatable subtransactions and the pivot subtransaction, which are value dependent on $G_{ij}$.

Therefore, the combination of global committability with compensation can greatly extend the class of global transactions for which the GTM can preserve atomicity without violation of local autonomy. A detailed discussion of such combination of the forward and backward recovery approaches is beyond the goal of this paper and thus is not presented here.

# 7 Summary

Reliable transaction management in the MDBS environment has been recognized as a substantial and as yet unresolved issue in those cases where the component local database systems do not support prepare-to-commit states. We have here advanced an approach to the atomic commitment of global transactions which uses the forward recovery approach to preserve the atomicity of global transactions in the MDBS environment. The approach provides an alternative to the 2PC protocol for reliable global transaction management in MDBSs through the resubmission of aborted global subtransactions.

Global committability employs the retry technique and the semantics of global transactions to control the commitment order of global subtransactions at the global level. Value dependency relationships among global subtransactions present the major obstacle to ensuring that each global subtransaction will be retriable without violating multidatabase consistency. As no restrictions are placed on local sites, local autonomy is fully preserved. The application of global committability is based upon the assumption that serializability is maintained on the execution of local and global transactions.

Global committability can be extended by combining it with the compensation, a novel technique of the backward recovery approach. This combination can significantly remedy the drawbacks of each individual approach. Value dependency relationships can be permitted to exist among the global subtransactions of each global transaction, making possible an extended global transaction model comparable to that proposed in [13]. An extensive class of global transactions that can be executed in the error-prone MDBS environment without violation of autonomy can therefore be formulated.

## References


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

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

[3] Y. Breitbart, D. Georgakopolous, M. Rusinkiewicz, and A. Silberschatz. On Rigorous Transaction Scheduling. *IEEE Transactions on Software Engineering*, 17(9):954–960, 1991.

[4] 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.

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

[6] A. K. Elmagarmid and A. Zhang. Supporting the Decomposition of Global Transactions in Multidatabase Systems. Technical Report CSD-TR-93-037, Purdue University, July. 1993.

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

[8] D. Georgakopoulos. Multidatabase recoverability and recovery. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pages 348–355, Kobe, Japan, Apr. 1991.

[9] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. On serializability of multidatabase transactions through forced local conflicts. In *Proceedings of the 7th Intl. Conf. on Data Engineering*, pages 314–323, Kobe, Japan, Apr. 1991.

[10] E. Levy, H. Korth, and A. Silberschatz. A theory of relaxed atomicity. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Aug. 1991.

[11] 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.

[12] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. Ensuring transaction atomicity in multidatabase systems. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1992.

[13] 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.

[14] P. Muth and T. Rakow. Atomic commitment for integrated database systems. In *Proceedings of the 7th Intl. Conf. on Data Engineering*, pages 296–304, Kobe, Japan, Apr. 1991.

[15] A. Silberschatz, M. Stonebraker, and J. Ullman. Database systems: Achievements and opportunities. *Communication of ACM*, 34(10):110–120, 1991.

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

[17] A. Zhang and J. Jing. On structural features of global transactions in multidatabase systems. In *Proceedings of the Third International Workshop on Interoperability in Multidatabase Systems*, pages 199–206, Vienna, Austra, April 1993.