On Structure Features of Global Transactions in Multidatabase Systems

Aidong Zhang

Jin Jing

Report Number:
92-042
ON STRUCTURAL FEATURES OF GLOBAL TRANSACTIONS IN MULTIDATABASE SYSTEMS

Aidong Zhang
Jin Jing

CSD-TR-92-042
July 1992
On Structural Features of Global Transactions in Multidatabase Systems *

Aidong Zhang and Jin Jing
Department of Computer Science
Purdue University
West Lafayette, IN 47907
{azhang,jing}@cs.purdue.edu

Abstract

Supporting the isolation and atomicity of global transactions in multidatabase systems has become an increasingly difficult task. Local autonomy requires that the maintenance of isolation and atomicity as an execution condition of global transactions must also hold when such execution is interleaved with local transactions. This paper proposes the utilization of both the syntactic and semantic knowledge of global transactions to formulate such a condition. An enriched combination of two theories, hybrid serializability and global committability, is presented as a vehicle for the maintenance of the isolation and semantic atomicity of global transactions. We then explore the structural features of global transactions which provide the necessary conditions for supporting this theory. As a result, we raise a new research issue for the management of global transactions. Instead of seeking restrictions on local transaction management systems, we have shown that placing conditions on global transactions to accommodate the autonomous local environment is a valid approach to preserving local autonomy.

1 Introduction

An effective transaction management system serves to maintain the ACIDity properties (i.e., atomicity, consistency, isolation, and durability) of transactions. In multidatabase systems (MDBSs), a global transaction management system must be superimposed on the

* Sponsored by the Indiana Business Modernization and Technology Corporation (BMT), a PYI Award from NSF under grant IRI-8857952, grants from the AT&T Foundation, Bellcore, Tektronix, SERC, Mobil Oil and Bell Northern Research.
pre-existing local transaction management systems. The difficulties of global transaction management in multidatabase systems arise primarily from the constraints posed by the autonomy of local database systems. Various aspects of autonomy, such as design, execution, and control, have been studied in [GMK88, BS88, Ve90], and their impact on global transaction management is discussed in [DEK90].

To preserve the isolation of global transactions without violating local autonomy, a global concurrency controller must be a component of the global transaction management system. This controller maintains the serializability of execution of global transactions while allowing such executions to interleave with the globally uncontrolled execution of local transactions at local sites. Since global subtransactions are received by the local transaction management systems and treated as local transactions, the global concurrency controller must incorporate the serialization orders in a manner which is consistent with the local level.

The traditional definition of atomic commitment has been shown [LKS91, MEKSA92, MRKS92] to be improper for the preservation of the atomicity of global transactions in a multidatabase environment. The concept of semantic atomicity, which was originally proposed [GM83, GMS87] to address the atomicity of long-running transactions, has been shown [LKS91] to be useful also in the multidatabase environment. With this concept, the global subtransactions of a global transaction are allowed to commit (or abort) unilaterally at local sites. Semantic atomicity guarantees that if all global subtransactions commit, then the global transaction commits; otherwise, either all tentatively committed global subtransactions are compensated (backward approach) or all aborted global subtransactions are retried (forward approach) until they commit. Compensation requires that the local transactions to be submitted must be commutative with the executing global subtransaction before its commitment or its compensating transaction may be executed. The retrial approach stipulates that no value dependencies [DE89] may exist between the subtransactions of a global transaction [MRKS92].

The theories of hybrid serializability and global committability have been independently developed [ZE93a, EJK91] as approaches to the maintenance of the isolation and semantic atomicity of global transactions in MDBSs. Hybrid serializability defines an optimal cor-
rectness criterion on global transaction scheduling without violating local autonomy, while global committability incorporates the semantic knowledge of global transactions into the retrial approach to extend the class of global transactions that can be executed in an error-prone multidatabase environment. In this paper, we will present a general principle of global transaction management which effectively combines both hybrid serializability and global committability. We will then develop the structural features required of a global transaction to accommodate the application of this combined theory.

The body of this paper is organized as follows. Section 2 introduces the system model and defines the terminology to be used. In Sections 3, the theoretical approach to such a combination of hybrid serializability and global committability is presented. Section 4 develops the structural features of global transactions necessary to the application of this approach. A final discussion is found in Section 5.

2 The System Model and Notations

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 transaction manager (GTM). Global transactions are submitted to the GTM, while local transactions are submitted to LDBSs. Figure 1 depicts the model.

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

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 \textit{commit} and \textit{abort} termination operations, and \(r(x)\) and \(w(x)\) are \textit{read} and \textit{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 \textit{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 [BH87, Had88]. A local transaction is a transaction that accesses the data items at a single local site. A global transaction is a set of global subtransactions, where each global subtransaction is a transaction accessing the data items at a single local site. The global transaction $G_i$ consists of a set of global subtransactions $\{G_{ij1}, G_{ij2}, \ldots, G_{ijr}\}$, where the subtransaction $G_{ijl}$ $(1 \leq l \leq r)$ is a transaction accessing $LDBS_{jl}$. A transaction $T_i$ refers to either a local or global transaction, and $OPT_i$ denotes the set of operations contained in $T_i$.

Two local transactions $T_i$ and $T_j$ conflict, denoted $T_i \prec T_j$, if there exist conflicting operations $o_i$ and $o_j$ such that $o_i \in OPT_i$ and $o_j \in OPT_j$. Let $DT$ denote the set of data items.
that transaction $T$ accesses. Two local transactions $T_i$ and $T_j$ are fully sharing, denoted $T_i \subseteq (\text{or } \supseteq) T_j$, if $D_{T_i} \subseteq D_{T_j}$ (or $D_{T_i} \supseteq D_{T_j}$).

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 be found in [BH87, 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 $S_g$ is $S$ restricted to the set $G$ of global transactions in $S$. A lower case $s$ refers to either a local or global schedule.

We say that a schedule $s$ is serial if the operations of different transactions in $s$ are not interleaved. We say that the execution of $T_1$ precedes the execution of $T_2$ in 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 $o_1 \prec_s o_2$ if operation $o_1$ is executed before operation $o_2$ in the schedule $s$. We denote $T_1 \prec_s T_2$ if, for transactions $T_1$ and $T_2$ in $s$ and any operation $o_1 \in T_1$ and any operation $o_2 \in T_2$, $o_1 \prec_s o_2$.

Let $s$ be a schedule and $C(s)$ be $s$ restricted to the committed transactions in $s$. We say $s$ is serializable$^1$ if there exists a serial schedule $s'$ and $C(s)$ is (conflict) equivalent$^2$ to $s'$. 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 $T_1 \prec_s T_2$ if $T_1$ precedes $T_2$ in the serialization order of $s$.

3 Maintaining the Isolation and Semantic Atomicity of Global Transactions

In this section, we shall present the fundamental principles that guide the maintenance of the isolation and semantic atomicity of global transactions in an MDBS environment without violation of local autonomy.

---

$^1$In this paper, serializability refers to conflict serializability.

$^2$See the definition given in [BH87, Had88].
3.1 The General Principle

We shall first present a formal definition of the maintenance of the isolation and semantic atomicity of global transactions without violation of local autonomy.

Let \( S \) be a global schedule and \( S_G \) be \( S \) restricted on the set \( G \) of global transactions in \( S \). At the global level, the GDBS controls the submission of global transactions to local sites, and consequently, the execution of global transactions. However, it must maintain the serializability of the global schedule \( S \) and the semantic atomicity of global transactions in \( S \), with \( S \) being the mixture of operations in both global and local transactions. At the local level, each LDBS freely executes both local transactions and global subtransactions, as long as local correctness criteria are preserved. It is intuitively obvious that we must determine those properties of global subschedules which (1) maintain the isolation and semantic atomicity of global transactions and which (2) must hold even if the operations of local transactions not in \( G \) are appended during the execution of a global transaction.

Formally, we say that a property \( P \) of global subschedule \( S_G \) is \textit{prefix local extensible} if, whenever it holds for \( S_G \), it also holds when any \( S_G' \leq S_G \) (\( S_G' \leq S_G \) denotes "\( S_G' \) is a prefix of \( S_G \)") is appended to any number of operations of local transactions that follow the correctness criteria for execution of transactions at local sites.

Our goal is to seek such prefix local extensible properties of global subschedules which maintain the isolation and semantic atomicity of global transactions. The hybrid serializability and global committability proposed in [ZE93a, EJK91] are two such properties.

3.2 Hybrid serializability

As stated in [GPZ86], global serializability cannot be maintained generally in MDBSs if a global transaction has more than one subtransaction at a given local site. Thus, in this section, we assume that each global transaction has at most one subtransaction at each local site. Since a global schedule is the combination of all local schedules, the global serialization order must inherit local serialization orders.

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. Let $S$ be a global schedule. It has been defined in [MRB+92] that, if 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 $O$, then $S$ is serializable. Therefore, the maintenance of global serializability can be reduced to the synchronization at all local sites of the relative serialization orders of the local subtransactions of each global transaction. However, due to the constraints of local autonomy, the GDBS may not be able to generate all global schedules satisfying the above sufficient condition. We need to identify alternative correctness conditions to be placed on global subschedules to provide sufficient conditions for the GDBS to maintain global serializability without placing restrictions at local sites.

The discussion of local indirect conflicts [DE89, BGRS91, GRS91] 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. This observation is illustrated by the following example.

**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:

- $G_1 : w_{g_1}(a)r_{g_1}(b)$
- $G_2 : r_{g_2}(a)r_{g_2}(c)$

Let $L_{2,1}$ be a local transaction submitted at local site $LS_2$:

- $L_{2,1} : w_{L_{2,1}}(b)w_{L_{2,1}}(c)$.

Let $S_1$ and $S_2$ be local schedules:

- $S_1 : w_{g_1}(a)r_{g_2}(a)$, $S_2 : w_{L_{2,1}}(b)r_{g_1}(b)r_{g_2}(c)w_{L_{2,1}}(c)$,

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 $G_{22} \rightarrow L_{2,1} \rightarrow G_{12}$. The serialization order of global subtransactions at local site $LS_2$ is not consistent with their execution order; this arises from the indirect conflict of $G_{22}$ with $G_{12}$ (since $r_{g_2}(c)$ conflicts with $w_{L_{2,1}}(c)$ and $w_{L_{2,1}}(b)$ conflicts with $r_{g_1}(b)$). However, if $w(b)$ is appended to $G_2$; that is, $G_2 = r_{g_2}(a)r_{g_2}(c)w_{g_2}(b)$;
then at local site LS2 after \( w_{L2,1}(b)r_{g1}(b) \) is scheduled, \( w_{L2,1}(c) \) would have to be scheduled before \( r_{g2}(c) \) to maintain local serializability. Hence, the correct schedule for \( S_2 \) is:

\[
S_2 : w_{L2,1}(b)r_{g1}(b)w_{L2,1}(c)r_{g2}(c)w_{g2}(b),
\]

which implies \( G_{12} \prec_{Sr} G_{22} \). The conflicting relationship between global subtransactions \( G_{12} \) and \( G_{22} \) here imposes an indirect effect on the local scheduling.

Suppose we instead append \( r(b) \) to \( G_2 \); that is, \( G_2 = r_{g2}(a)r_{g1}(c)r_{g2}(b) \). Then at local site LS2, after \( w_{L2,1}(b)r_{g1}(b) \) is scheduled, \( w_{L2,1}(c) \) would still have to be scheduled before \( r_{g2}(c) \) to maintain local serializability. Hence, the correct schedule for \( S_2 \) is:

\[
S_2 : w_{L2,1}(b)r_{g1}(b)w_{L2,1}(c)r_{g2}(c)r_{g2}(b),
\]

which implies \( G_{12} \prec_{Sr} G_{22} \). The fully sharing relationship (that is, \( D_{G_{12}} \subseteq D_{G_{22}} \)) between global subtransactions \( G_{12} \) and \( G_{22} \) here imposes an indirect effect on the local scheduling, 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 both \( G_{12} \) and \( G_{22} \) such that \( G_{22} \prec_{Sr} L \prec_{Sr} G_{12} \).

Intuitively, we observe that controlling the order of either the conflicting operations or the sharing operations in two global transactions may control the serialization order of these two global transactions at local sites. Provided that local serializability is maintained, the operations of global transactions then interleave with the operations of local transactions. In [ZE93a], we proposed a new correctness criterion, called hybrid serializability, based on the above observation. Let \( G_k \) denote the set of global subtransactions of \( G \) at local site \( LS_k \). The central principle can be stated as follows:

**Definition 1 (Hybrid dependency)** A set \( T \) of local transactions is hybrid dependent if there is a total order \( T_{i1}, T_{i2}, \ldots, T_{in} \) on \( T \) such that \( T_{i1} \triangleleft T_{i2} \triangleleft \cdots \triangleleft T_{in} \) where \( \triangleleft \in \{ \prec, \leq, \geq \}^3 \).

A set \( G \) of global transactions is hybrid dependent if there is a total order \( O \) on \( G \) such that for all \( k, 1 \leq k \leq m, G_k \) is hybrid dependent in an order consistent with \( O \).

**Definition 2 (Hybrid equivalence)** Two global subschedules \( S_G \) and \( S'_G \) of global schedule

---

\(^3\)We consider that \( \preceq \) has higher priority to be chosen than \( \leq \) (or \( \geq \)); in other words, if two transactions \( T_i \) and \( T_j \) have both \( T_i \preceq T_j \) and \( T_i \leq \) (or \( \geq \)) \( T_j \) properties, then \( T_i \preceq T_j \) will be chosen in the hybrid dependency instead of \( T_i \leq \) (or \( \geq \)) \( T_j \). Both \( \leq \) and \( \geq \) have the same priority.
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 dependent in a total order $O$ on $\mathcal{G}$ and for any $G_i$ directly preceding $G_j$ in $O$, the following conditions are satisfied for all integer $k$ $(1 \leq k \leq m)$:

- if $G_{ik} \preceq G_{jk}$, then for all conflicting operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}, o_{ik} \prec_{eo}^\mathcal{G} o_{jk}$ and $o_{ik} \prec_{eo}^\mathcal{G} o_{jk}$; or
- if $G_{ik} \subseteq G_{jk}$ (or $G_{ik} \supseteq G_{jk}$), then for all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}, o_{ik} \prec_{eo}^\mathcal{G} o_{jk}$ and $o_{ik} \prec_{eo}^\mathcal{G} o_{jk}$.

Definition 3 (Hybrid serializability) A global subschedule is hybrid serializable if and only if it is hybrid equivalent to a serially global subschedule.

Note that hybrid serializability is stronger than serializability; that is, hybrid serializability implies serializability. The following theorem is proven in [ZE93a]:

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

Following Theorem 1, global serializability can be achieved at the global level by controlling the execution order of the set of global transactions which is hybrid dependent. In addition, only the operations that determine the hybrid dependency relationship need to be ordered. The property of hybrid serializability on global subschedules is prefix local extensible. In other words, if the GDBS maintains hybrid serializability on global subschedules, then the interleaving of the operations of global and local transactions at local sites will not change the hybrid serialization order of global transactions.

### 3.3 Global Committability

We shall now address the problem of maintaining the semantic atomicity of global transactions. We present a more efficient global-level application of the retrial approach achieved by controlling the commit order of global subtransactions.
The retrial approach to maintaining the semantic atomicity of global transactions allows the retrial of an aborted global subtransaction. It is assumed that the subtransaction will commit after it is retried a sufficient number of times. Some difficulties may arise, however, if one global subtransaction has a value dependency relationship with another global subtransaction. For instance, assume that a value written by \( G_{il} \) at local site \( LS_l \) is dependent on a value read by \( G_{ij} \) at local site \( LS_j \). If \( G_{il} \) commits and \( G_{ij} \) aborts, then \( G_{ij} \) must be retried. However, local transactions may be executed after \( G_{ij} \) is aborted but before it is retried. Consequently, the data read from the original execution of \( G_{ij} \) and from its retrial might be different. To overcome this inconsistency, an approach is proposed in [MRKS92] that stipulates that no value dependencies may exist between the subtransactions of a global transaction. In this formulation, the execution of a global transaction at one local site is independent of its execution at other sites.

To extend the class of global transactions that can be executed in the MDBS environment, the concept of global committability is proposed in [EJK91]. This approach utilizes the semantic relationships among the global subtransactions of a global transaction to control the commit order of global subtransactions at local sites. For instance, let \( G_i = \{G_{i1}, G_{i2}, \ldots, G_{im}\} \) be a global transaction, where \( G_{ij} \) \((1 \leq j \leq m)\) is a global subtransaction accessing \( LDBS_j \). Assume the value read by \( G_{ij} \) and \( G_{ik} \) determines a value written by \( G_{il} \) \((1 \leq j, k, l \leq m)\). The general retrial approach is not applicable to \( G_i \), since \( G_{ij} \) or \( G_{ik} \) is not retriable after \( G_{il} \) commits. However, global committability specifies that \( G_{il} \) must commit after \( G_{ij} \) and \( G_{ik} \) have committed. Thus, every global subtransaction is retriable.

We now present an enriched global committability approach which incorporates the advantageous features of hybrid serializability. Without loss of generality, let \( G_i \) consist of the global subtransactions \( G_{i1}, G_{i2}, \ldots, G_{im} \), where \( G_{ij} \) is the global subtransaction at \( LS_j \). We say that \( G_{ij} \) is value-dependent on \( G_{ij_1}, \ldots, G_{ij_{i-1}} \) \((1 \leq j_1, \ldots, j_i \leq m)\), denoted \( \{G_{ij_1}, G_{ij_2}, \ldots, G_{ij_{i-1}}\} \rightarrow_v G_{ij_i} \), if there exists an update operation of \( G_{ij_i} \) such that the value written by this update operation is a function of the values read by \( G_{ij_1}, \ldots, G_{ij_{i-1}} \). The value dependencies of global subtransactions of a global transaction 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 $G_{ij_1}$ is one of the elements upon which $G_{ij_2}$ is value-dependent.

Let $G = \{G_1, G_2, \ldots, G_n\}$ be the set of global transactions in the global schedule $S$ and let $G$ be hybrid dependent in a total order $O$ on $G$. Assume that each $VDG(G_i)$ ($1 \leq i \leq n$) is acyclic. The enriched global commitment (GC) protocol is formulated as follows:

1. (Intra-Committability) For each global transaction $G_i$, if $\{G_{ij_1}, \ldots, G_{ij_{n-1}}\} \rightarrow_v G_{ij_1}$, then $G_{ij_1}, G_{ij_2}, \ldots, G_{ij_{n-1}}$ must commit before $G_{ij_1}$ commits.

2. (Inter-Committability) For the set $G$ of global transactions, at each local site, the commit order of global subtransactions in $G_k$ is consistent with $O$.

Condition 1 above requires that a global subtransaction can only commit after all global subtransactions upon which it is value-dependent have committed. Thus, it ensures that each global subtransaction is retriable, even if there exist value dependencies among global subtransactions. Consequently, global transactions are semantically atomic, provided that each global subtransaction commits after it is retried a sufficient number of times. Condition 2 ensures that, when two or more global transactions run concurrently, hybrid serializability is maintained, even in a situation in which global transactions may abort.

The following theorem is proven in [EJK91]:

**Theorem 2** The GC commitment protocol preserves the semantic atomicity of global transactions.

The GC commitment protocol satisfies the prefix local extensible property. In other words, every global transaction remains globally committable while the operations of local transactions are appended to a prefix $S'_G$ of a global subschedule $S_G$. Since the GC commitment maintains the commit order as identical to the hybrid serialization order, hybrid
serializability is maintained in the error-prone multidatabase environment. Hence, we have the following corollary:

**Corollary 1** Hybrid serializability and the GC commitment protocol preserve the isolation and semantic atomicity of global transactions.

Difficulties may arise if the value dependency graph of a global transaction is not acyclic; the GC commitment protocol is not applicable in these instances. We will address this situation in the next section.

4 **The Structural Features of Global Transactions**

The structural features of global transactions are considered in reference to those properties relevant to the theory of isolation and semantic atomicity of global transactions discussed in the Section 3. Initially, it is obvious that the set of global transactions submitted must be hybrid dependent to allow the application of hybrid serializability in the GDBS. We now consider those additional structural features of global transactions necessary for the application of the GC commitment protocol.

4.1 **Global Subtransaction Decomposition**

A necessary condition for the application of GC commitment protocol is the association of each global transaction with an acyclic value dependency graph, a condition which may not be true for all global transactions. We shall now discuss the decomposition of global subtransactions to maintain the acyclicity of their value dependency graphs.

A global subtransaction may initially be decomposed into two global subtransactions. Following the lowest-level transaction model (rather than the multi-level model), a decomposed global subtransaction must possess certain properties.

Let $X$ be a transaction, $X\text{ReadSet}$ denote the set of data items $X$ reads, and $X\text{WriteSet}$ denote the set of data items $X$ writes.
In a manner comparable to the splitting of transactions proposed in [PKH88], the following three properties are considered as fundamental to the decomposition of a global subtransaction $G_{ij}$ (which is the global subtransaction of $G_i$ at local site $LS_j$) into two global subtransactions $A$ and $B$, with $A$ preceding $B$ in the execution:

- $AWriteSet \cap BWriteSet \subseteq BWriteLast$
  That is, if $A$ and $B$ write to same data items, then these data items are written last by $B$. This property prevents $A$ overwriting $B$'s output.

- $AReadSet \cap BWriteSet = \emptyset$
  That is, $A$ will not read from $B$.

- $BReadSet \cap AWriteSet = ShareSet$
  That is, if $A$ writes to and $B$ reads from an identical data item, then $B$ reads from $A$.

The above three properties define the necessary conditions for the decomposed global subtransaction $A$ to be executed before the decomposed global subtransaction $B$. If the value dependency graph of the decomposed global transaction is still cyclic, then further redecomposition of certain global subtransactions of the global transaction must be implemented.

The decomposition thus redress the cyclicity of the value dependency graphs of global transactions. A simple example is shown below to explain.

**Example 2** Suppose $VDG(G_i)$ of $G_i = \{G_{i1}, G_{i2}\}$ is:

![Diagram of $G_{i1}$ and $G_{i2}$]

If one step decomposition on $G_{i1}$ produces:
then the graph of $VDG(G_i)$ is acyclic. Consequently, the order of the intra-committability of $G_i$ is $G'_{i1} \rightarrow G_{i2} \rightarrow G''_{i1}$. If the above decomposition would not result in an acyclic graph, then further decomposition needs to be pursued.

A global transaction is considered to be decomposable if its value dependency graph becomes acyclic after sufficient steps of decomposition which satisfies the above three properties. If a global transaction is decomposable, then its semantic atomicity can be preserved by means of the GC commitment protocol. Hence, we have the following corollary:

**Corollary 2** The GC commitment protocol maintains the semantic atomicity of decomposed global transactions.

Unfortunately, the above three conditions are not sufficient to allow the decomposition to be used in an MDBS environment since decomposition may cause each global transaction to have more than one global subtransaction at a local site. As a consequence, global serializability may not be maintained. For instance, in Example 2, the LDBS at $LS_1$ will consider $G'_{i1}$ and $G''_{i1}$ to be two different local transactions. Even though the GDBS can control that $G'_{i1}$ is serialized directly before $G''_{i1}$, there may be a local transaction $L$ which is locally serialized between $G'_{i1}$ and $G''_{i1}$, that is, $G'_{i1} \prec_{sr} L \prec_{sr} G''_{i1}$. Such cases cannot be prevented by the GDBS. This situation may create the following difficulties:

- From the local point of view, $L$ may read inconsistent data from $G'_{i1}$, or the value written by $L$ will be lost by the overwriting of $G''_{i1}$; and

- From the global point of view, global serializability cannot be maintained.

Thus, the correctness of the concurrent execution of decomposed global transactions at local sites needs to be reconsidered.
4.2 Global Consistency

We shall now discuss the effect of the decomposition of global transactions on global serializability.

In general, a database state is defined as mapping every data item to a value of its domain. The database integrity constraints are defined as a subset of all the possible database states, and a database state is consistent if it belongs to that subset [Pap86, MRKS91]. These integrity constraints are usually classified into explicit integrity constraints and implicit integrity constraints. The explicit integrity constraints are those constraints to be statically stated in the database system; and the implicit integrity constraints include the rest part of the integrity constraints. Since each decomposed global subtransaction is considered as an independent local transaction at a local site, subtransactions must satisfy the consistency property of a transaction. In other words, each decomposed global subtransaction must map a consistent state to another consistent state. Let us consider the decomposition of $G_{ij}$ into two global subtransactions $A$ and $B$. The following properties are additional conditions which are sufficient for the decomposed global subtransactions to preserve their consistency:

- **Value-Independency**: There must be no data exchange between $A$ and $B$.
- **Write-Independency**: $A$ and $B$ must not write simultaneously to the data items over which an explicit integrity constraint is defined.
- **Read-Before-Write**: $B$ must read a data item before it may write to the data item.

The **Value-Independency** condition ensures that the execution of $A$ preserves the implicit integrity constraints; the **Write-Independency** condition ensures the execution of $A$ preserves the explicit integrity constraints; the **Read-Before-Write** condition ensures that $B$ will not destroy a consistent state of the database.

In Example 2, the decomposition is not value-independent since $G'_{i1}$ is indirectly value-dependent on $G''_{i1}$ through $G_{i2}$. If the further decomposition on $G_{i2}$ produces:
then, $G'_{i1}$ is value-independent on $G''_{i1}$; and $G'_{i2}$ is value-independent on $G''_{i2}$. We say that a global transaction is strongly decomposable if its value dependency graph becomes acyclic after sufficient steps of decomposition which satisfies all six properties given in Sections 4.1 and 4.2.

Note that the decomposition defined here is different from *sagas* [GMS87], which decomposes a transaction into several independent transactions. In this paper, the decomposed global subtransactions still belong to their original global transaction and the decomposed global subtransactions of a global transaction in different local sites may have value dependencies.

We define a schedule as *correct* if it preserves the database integrity constraints. If the above three additional conditions are satisfied by the decomposed global subtransactions, then the global schedules are correct, provided that the global subschedules are hybrid serializable and the local schedules are serializable. In order to apply hybrid serializability to decomposed global transactions, the hybrid dependency order must be forced for the decomposed global subtransactions, and these hybrid dependency orders must also be maintained through hybrid serializability. The following simple case illustrates the principle.

Let $G_1 = \{G_{11}, G_{12}\}$, $G_2 = \{G_{21}, G_{22}\}$, and $G_1$ precede $G_2$ in the hybrid dependency order. Suppose that $VDG(G_1)$ is cyclic, $VDG(G_2)$ is acyclic, and the strongly acyclic decomposition of $G_1$ is $\{G'_{11}, G''_{11}, G'_{12}, G''_{12}\}$. Hybrid serializability determines that $G'_{11} \prec_{sr} G''_{11} \prec_{sr} G_{21}$ and $G'_{12} \prec_{sr} G''_{12} \prec_{sr} G_{22}$. At local site $LS_1$, there exist a local transaction $L_1$ such that $G'_{11} \prec_{sr} L_1 \prec_{sr} G''_{11} \prec_{sr} G_{21}$, which results in a non-globally serializable schedule of $L_1$, $G_1$ and $G_2$. The GDBS cannot prevent such cases without violating local autonomy. However, global serializability can be relaxed, and, from certain point of view, the database consistency is guaranteed. Since $L_1$, $G'_{11}$ and $G''_{11}$ are all consistent transactions at $LS_1$, we
can consider \( L_1 \) as a part of the original global subtransaction \( G_{11} \). That is,

![Diagram of G11, Gi1, L1, Gi1]

Though the execution of \( G_1 \), with local transactions serialized between its decomposed global subtransactions, may result in a different outcome from the execution of an undecomposed transaction, it nonetheless maintains database consistency. Hence, all local transactions which are serialized between decomposed global subtransactions can be omitted in the consideration of global serializability.

We say that a global schedule \( S \) is pseudo-serializable if \( S \) is serializable without considering the local transactions which are serialized in the middle of two decomposed global subtransactions. As explained above, pseudo-serializable global schedules preserve the global consistency of the database if each global transaction is strongly decomposable. Thus, we have the following theorem:

**Theorem 3** Let \( S \) be a global schedule and \( G \) be the set of committed and strongly decomposed global transactions in \( S \). If \( S_G \) is hybrid serializable, then the local serializability of \( S_k \) (for \( k=1,\ldots,m \)) implies that \( S \) is correct.

**Proof:** By the results presented in [ZE93a, ZE93b], \( S \) is serializable without considering the local transactions which are serialized in the middle of two decomposed global subtransactions. Furthermore, the property of strong decomposition of global transactions ensures that no integrity constraint may be violated with local transactions being serialized in the middle of two decomposed global subtransactions. Thus, \( S \) is correct. \( \Box \)

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

If the multi-level model [WS91, Wei91] is applied in the decomposition of global transactions, the above properties may be relaxed slightly by means of the semantics of indi-
vidual operations. For instance, suppose $X = \text{Withdraw}(20, d) \text{Withdraw}(10, d)$\(^4\). Since $\text{Withdraw}(20, d)$ and $\text{Withdraw}(10, d)$ are commutative, we can decompose $X$ into $A = \text{Withdraw}(10, d)$ and $B = \text{Withdraw}(20, d)$, and $A$ is executed before $B$. Hence, the first property in Section 4.1 is relaxed; other properties can be similarly relaxed. Thus, the semantic knowledge of a global transaction can aid in a more efficient decomposition of global transactions.

5 Discussion

There has recently been a large volume of research directed towards the development of new approaches to transaction management in a multidatabase environment. The most successful proposals have concentrated primarily on defining restrictions on local transaction management in order to obtain global properties [BGRS91, BS92]. If the pre-existing local transaction management systems satisfy these restrictions, then these theories are applicable.

In this paper, we have addressed the restrictions on the global transaction management necessary to maintain the isolation and atomicity of global transactions in a multidatabase environment. The structural features of global transactions have been formalized in order to accommodate the demand for autonomy on the part of integrated components in a multidatabase environment. The lack of global-level knowledge regarding these autonomous components requires that we consider the placing of restrictions on global transaction management. Our ultimate goal has been to determine the most relaxed restrictions applicable to global transaction management maintaining the ACIDity properties of both global and local transactions.

Hybrid dependency must be considered as the fundamental structural feature of global transactions necessary for global transaction management to operate without violating local autonomy while maintaining global database consistency. Hybrid dependency permits the application of hybrid serializability, the optimal correctness criterion, to the execution of global transactions. Additional structural features of global transactions are also stip-

\(^4\)Withdraw(20, d) indicates the subtraction of 20 from data item d.
ulated as necessary conditions for the application of the GC commitment protocol. The application of hybrid serializability and global committability allow local autonomy to be fully maintained. We have seen that the characteristics (both syntactic and semantic) of global transactions can be constructively applied toward the preservation of the isolation and semantic atomicity of global transactions.

Enforcing the formulated structural features on global transactions is not a trivial task. A ticket method [GRS91] is proposed to force conflicts among all global transactions, thus implementing a strong condition of hybrid dependency. An appending operation method [ZE93a] is also used to force hybrid dependency on global transactions. Further research is in progress on these approaches.

The work presented here is therefore only a first step; much development remains on both the theoretical and practical aspects of enforcing the structural features of global transactions.

Acknowledgement

The authors have benefited greatly from discussions with Dr. Ahmed Elmagarmid, who is the head of the InterBase project. We thank him for his contributions to and critique of this paper. We also appreciate the financial support which he has provided, without which this work could not be pursued.

References


