Scheduling with Compensation in Multi-database Systems

Aidong Zhang
Bharat Bhargava

Purdue University, bb@cs.purdue.edu

Report Number:
93-063
SCHEDULING WITH COMPENSATION IN MULTIDATABASE SYSTEMS

Aidong Zhang
Bharat Bhargava

CSD TR-93-063
October 1993
(Revised March 1994)
Scheduling with Compensation in Multidatabase Systems

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

Abstract

A multidatabase system integrates a set of autonomous database systems to provide global database functions. This paper investigates a correctness criterion for the execution of local and global transactions in the error-prone multidatabase environment. The compensation technique is used to preserve the semantic atomicity of global transactions. We examine in detail the effects created by the value dependencies present among the subtransactions of a global transaction on concurrency control. This investigation assumes only the serializability and recoverability of local database systems. A global transaction scheduling mechanism is grounded upon the proposed theory.

1 Introduction

A multidatabase system integrates a set of autonomous database systems to provide global database functions. In a multidatabase system (MDBS), transaction management is handled at both the global and local levels. As a confederation of pre-existing local databases, the overriding concern of any MDBS must be the preservation of local autonomy [Lit86, GMK88, BS88, Pu88, Veig0]. This is accomplished through the superimposition of a global transaction manager (GTM) upon a set of local database systems (LDBSs). Global transactions are submitted to the global transaction manager, where they are parsed into a set of global subtransactions to be individually submitted to local transaction management systems. At the same time, local transactions are directly submitted to the local transaction management systems. Each local transaction management system maintains the correct execution of both
local and global subtransactions at its site. It is left to the global transaction manager to maintain the correct execution of global transactions.

The preservation of the atomicity and isolation of global transactions is fundamental in achieving the correct execution of global transactions. Preserving the atomicity or semantic atomicity [GM83] of global transactions in multidatabase systems has been recognized as an open and difficult issue [SSU91]. The traditional two-phase commit protocol (2PC) developed in distributed database environments has been shown [LKS91, SKS91, MR91] to be inadequate to the preservation of the atomicity of global transactions in the multidatabase environment. For example, some local database systems may not support a visible prepare-to-commit state, in which a transaction has not yet been committed but is guaranteed the ability to commit. In such situations, a local database system that participates in a multidatabase environment may unilaterally abort a global subtransaction without agreement from the global level. The concept of compensation, which was proposed [GM83] to address the semantic atomicity of long-running transactions, has been shown [LKS91] to be useful in the multidatabase system environment. Using this technique, the global subtransactions of a global transaction may commit unilaterally at local sites. Semantic atomicity guarantees that if all global subtransactions commit, then the global transaction commits; otherwise, all tentatively committed global subtransactions are compensated.

An assessment of the potential of such a technique mandates a careful examination of the effect of compensation on the concurrency control of global transactions. In [KLS90], a formal analysis is presented of those situations in which a transaction may see the partial effect of another transaction before these partial effects are compensated. It is then proposed in [LKS91] that, to prevent an inconsistent database state from being seen in a distributed database environment, a global transaction should be unaffected by both aborted and committed subtransactions of another global transaction. This theory is termed isolation of recovery. A concurrency control correctness criterion, termed serializability with respect to compensation (SrIC), is further proposed in [MRKS92] to preserve database consistency in theMDBS environment throughout the execution of global transactions possessing no value dependencies among their subtransactions. This criterion prohibits any global transaction that is serialized between a global transaction $G_i$ and its compensating transaction $CG_i$ from accessing the local sites at which $G_i$ aborts. All these proposed approaches are inadequate to a situation in which value dependencies are present among the subtransactions of a global
transaction. Value dependencies, which specify data flow among the global subtransactions of each global transaction, are characteristics of many applications [ZNBB94]. For example, many applications involve data transfer among different local database sites, generating value dependencies among the subtransactions of a global transaction.

This paper develops a new correctness criterion for the execution of local and global transactions involving compensation. In this scenario, LDBSs are assumed to ensure serializability\(^1\) and value dependencies are permitted among the subtransactions of all global transactions. In the proposed correctness criterion, serializability is ensured among local transactions, global transactions, and the compensating transactions of partially committed global transactions. In addition, the partial effects of the committed subtransactions of a global transaction will not be seen by other global transactions until either the entire global transaction commits or the partial effects are compensated. Our primary concern is to guarantee multidatabase consistency while still achieving high concurrency in the execution of local and global transactions.

This paper is organized as follows. Section 2 introduces the multidatabase transaction model, while Section 3 discusses the characteristics of compensating transactions in the MDBS environment. Section 4 proposes a correctness criterion which combines compensation with serializability. In Section 5, a mechanism is developed which preserves the semantic atomicity and isolation of global transactions. Concluding remarks are provided in Section 6.

2 The System Model and Terminology

An MDBS consists of a set of \(\{LDBS_i, \text{ for } 1 \leq i \leq n\}\), where each \(LDBS_i\) is a pre-existing autonomous database management system on a set of data items at the local site \(LS_i\), superimposed on which is a global database management system (GTM). Figure 1 depicts the model.

We assume the availability of four basic transaction operations: \(r(x), w(x), c, \text{ and } a\), where \(c\) and \(a\) are commit and abort termination operations and \(r(x)\) and \(w(x)\) are read and write accessing operations in a local database. We shall alternatively use \(r(x, v)\) (or

\(^1\)In this paper, serializability refers to conflict serializability [BHGG87].
Flexible Transaction

Subtransaction

Local Transaction

Server 1

Local Database System 1

Local Database 1

Subtransaction

Local Transaction

Server n

Local Database System n

Local Database n

Figure 1: The multidatabase system model.

$w(x,v)$ to denote an operation which reads (or writes) a value $v$ from (or to) data item $x$. Two operations conflict with each other if they access the same data item and at least one of them is a write operation. In this paper, a transaction is a sequence of read and write operations followed by either a commit or an abort termination operation. In an MDBS environment, a local transaction is a transaction that accesses the data items at a single local site. A global transaction is a set of global subtransactions in which each global subtransaction is a transaction accessing the data items at a single local site. We assume that each global transaction has only one subtransaction at each local site [GPZ86]. $G_{ij}$ denotes a global subtransaction of $G_i$ accessing LDBS$_j$.

The operations of the different subtransactions of a global transaction are related according to their value dependencies. Let global transaction $G_i = \{G_{i1}, G_{i2}, \cdots, G_{in}\}$. $G_{ij}$ is value dependent on $G_{ij_1}, \ldots, G_{ij_{j-1}}$ ($1 \leq j_1, \ldots, j_t \leq n$), denoted $G_{ij_t} \rightarrow_v G_{ij_1}, G_{ij_t} \rightarrow_v G_{ij_2}, \ldots, G_{ij_{j-1}} \rightarrow_v G_{ij_t}$, if the execution of one or more operations in $G_{ij_t}$ is determined by the values read by $G_{ij_1}, \ldots, G_{ij_{j-1}}$. We assume that value dependencies are the only relationships present among the global subtransactions of a global transaction.

We shall use the term schedule to refer to a partial order of operations resulting from the
concurrent execution of a set of transactions. A local schedule is a schedule over both local transactions and global subtransactions which are executed at a local site. A global schedule is a schedule over all local and global transactions.

In an MDBS system, there are two types of integrity constraints: local integrity constraints are defined on data items at a single local site, while global integrity constraints are defined on data items at multiple local sites. A local database state, that is a mapping of every data item at the local site to a value of its domain, is consistent if it preserves local database integrity constraints. A multidatabase state, that is a mapping of every data item in the multidatabase to a value of its domain, is globally consistent if it preserves all integrity constraints defined in the MDBS environment.

Definition 1 (Schedule correctness) A schedule is correct if it preserves all integrity constraints that are defined in the database system and each transaction in $S$ reads only a consistent database state.

3 Compensating Transactions in Multidatabase Systems

Traditionally, we say that a transaction is compensatable if the effects of its execution can be semantically undone after commitment by executing a compensating transaction. Let a compensating transaction for a global subtransaction $G_{ij}$ be $CG_{ij}$. In addition to satisfy the traditional definition of compensation, $CG_{ij}$ for $G_{ij}$ must also be independent of the transactions that execute between $G_{ij}$ and $CG_{ij}$. That is, the effects of a compensatable global subtransaction $G_{ij}$ must be undoable regardless of any executions that have occurred between $G_{ij}$ and $CG_{ij}$. Such independence is not required in traditional transaction compensation, where the problem of the uncontrolled interleaving of local transactions does not arise. Local autonomy here requires that arbitrary local transactions must be executable while compensation is effected and no local transactions that saw the effects of compensated global subtransactions need to be rolled back. Thus, the application of compensation in the MDBS environment is more restrictive. The following example is illustrative:
Example 1 Consider an MDBS that has data item \(a\) in \(LS_1\) and \(b\) in \(LS_2\). Let the integrity constraint be \(a \geq 0\) and \(b \geq 0\). Suppose a global transaction that transfers half of the amount in \(a\) to \(b\) is executed:

\[
G_1 : r_{G_{12}}(b) r_{G_{11}}(a) w_{G_{12}}(b, b + 0.5 \times a) w_{G_{11}}(a, a - 0.5 \times a)
\]

In this example, \(G_{11}\) is compensatable, while \(G_{12}\) is not. The compensation of \(G_{12}\) may violate the integrity constraint \(b \geq 0\) if a local transaction which is executed between \(G_{12}\) and its compensating transaction takes the amount of \(b\). Note that both \(G_{11}\) and \(G_{12}\) are compensatable in the traditional distributed database environment, as long as the transactions that can be executed between a global subtransaction and its compensating transaction are commutative [KLS90] with the compensating transaction.

We assume that each compensating subtransaction \(CG_{ij}\) for a compensatable subtransaction \(G_{ij}\) can always commit. That is, it is guaranteed that any compensation initiated will complete successfully. This requirement, termed persistence of compensation, has been discussed in the literature [GM83]. In addition, a compensating global transaction \(CG_i\) for global transaction \(G_i\) is a separate global transaction from \(G_i\) and consists of global subtransactions that compensate the committed global subtransactions of \(G_i\).

We permit value dependencies to be present among the subtransactions of a global transaction, as long as each global subtransaction can be compensated at a single local site. As has been discussed in the literature [LKS91, MRKS92], we consider it to be impractical in the MDBS environment for the compensating transaction of a committed global subtransaction to be executed at multiple sites. However, given careful maintenance, a committed global subtransaction can still be compensated at a single local site, even though the effects of the global subtransaction may have spread to other sites. For instance, in Example 1, if \(a \geq 0\) and \(b \geq 0\) are relaxed, then both \(G_{11}\) and \(G_{12}\) are compensatable. Note that \(G_{12}\) is value dependent on \(G_{11}\). In order to prepare for the potential execution of the compensating transaction \(CG_{12} : r(b) w(b, b - 0.5 \times a)\) at \(LS_2\), a parameter which contains the value of \(a\) should be maintained at \(LS_2\) when \(G_{12}\) is executed. This parameter can then easily be passed to \(CG_{12}\) when it is to be executed.
4 Compensation Serializability

In this section, we presume that all global subtransactions are compensatable and that serializability is initially established as the correctness criterion for the execution of local and global transactions. The submission of a global subtransaction should be accompanied by the its compensating subtransaction. The effect of compensation on serializability is then carefully analyzed.

Clearly, a serializable global schedule preserves both local and global integrity constraints in the event that all local and global transactions individually preserve both local and global integrity constraints. However, local autonomy may cause local transactions to be totally unaware of global integrity constraints. In this instance, local transactions may preserve only local integrity constraints, while global transactions must preserve both local and global integrity constraints. In addition, since global subtransactions are considered as independent transactions at each local site, they must preserve at least local integrity constraints. We thus assume that local transactions preserve only local integrity constraints, while global transactions preserve both local and global integrity constraints. We further assume that no global integrity constraints may be placed on those data items that are updatable by local transactions. Otherwise, such updating may result in the violation of global integrity constraints. In such a situation, even a serializable global schedule would be incapable of maintaining multidatabase consistency.

All global subtransactions are permitted to commit unilaterally. Similar to [MRKS92], we define a global subtransaction in global schedule $S$ to be compensated-for if it has committed in $S$ and its effects need to be compensated. A global transaction $G_i$ in global schedule $S$ is compensated-for if it has compensated-for global subtransactions in $S$.

When a global subtransaction commits, the need for compensation has not yet been determined. If this subtransaction is eventually compensated, its results form the partial effects of a global transaction that may not be globally consistent. Clearly, local transactions can read such partial effects of a global transaction, because the execution of a global subtransaction always preserves local database consistency. Whether other global transactions should be allowed to read such partial effects of a global transaction is less immediately apparent.

A correctness criterion (SRC) has been proposed in [MRKS92] for the execution of global
transactions possessing no value dependencies among their subtransactions when compensation is involved. This criterion prohibits any global transaction that is serialized between a global transaction $G_i$ and its compensating transaction $CG_i$ to access the local sites at which $G_i$ aborts. However, in general, even though SRC is ensured in a global schedule $S$ and compensating transactions undo the effects of compensated-for global subtransactions in $S$, $S$ may still not preserve multidatabase consistency. The following example illustrates the situation:

Example 2 Consider an MDBS that has data item $a$, $b$ at $LS_1$ and data item $c$ at $LS_2$. Let the integrity constraints be $a > c$ and $b > c$. Let two global transactions $G_1$ and $G_2$ be:

$G_{11} : r(b)w(b, b - 1), \quad G_{12} : r(c)w(c, c - 1)$

$G_{21} : r(b)w(a, b)$.

Consider a global transaction $G_1$ that results from database state $a = 1, b = 1, c = 0$, where $G_{11}$ commits, $G_{12}$ aborts, and a global transaction $G_2$ executes after $G_1$. A compensable global transaction $CG_1 : r(b)w(b + 1)$, which is independent of $G_2$, undoes the effect of $G_{11}$. $G_1$, $G_2$, and $CG_1$ are serializable in the order $G_1 \rightarrow G_2 \rightarrow CG_1$. There are no data dependencies between subtransactions of $G_1$. The global schedule is SRC. However, the resulting database state, which is $a = 0, b = 1, c = 0$, is inconsistent.

We realize\(^2\) that inconsistencies such as those illustrated in Example 2 arise because compensating transaction $CG_1$ fails to restore database consistency after it executes. In order to avoid such situations, $CG_1$ must also undo any effects that may have been seen by other global transactions. Compensating transactions must therefore be dynamically constructed to take account any executions that have occurred after the commitment of the global subtransactions. Such considerations complicate the task of constructing compensation transactions.

In addition to the situation illustrated above, the SRC criterion is not applicable in instances in which value dependencies are defined on global transactions. The following example is illustrative:

Example 3 Consider an MDBS that has data item $a$ at $LS_1$, data item $b$ at $LS_2$, and data

\(^2\)As suggested by discussions with Dr. Sharad Mehrotra.
item c at LS3. Let the integrity constraints be \( a < c, b < c, \) and \( a = b. \) Let a global transaction \( G_1 \) consist of two subtransactions:

\[
G_{11} : r(a)w(a, a - 1), \quad G_{12} : r(b)w(b, b - 1).
\]

Let another global transaction \( G_2 \) be:

\[
G_{21} : r(a), \quad G_{22} : w(c, a + 1).
\]

Consider an execution of \( G_1 \) that results from database state \( a = 3, b = 3, c = 5, \) where \( G_{11} \) commits while \( G_{12} \) aborts and \( G_2 \) executes after \( G_1. \) A compensatable transaction \( CG_1 : r(a)w(a, a + 1) \), which is independent of \( G_2 \), then undoes the effect of \( G_{11}. \) \( G_{11}, G_2, \) and \( CG_{11} \) are serializable in the order \( G_{11} \rightarrow G_2 \rightarrow CG_{11}. \) \( G_2 \) does not access the local site where \( G_1 \) aborts. However, the resulting database state, which is \( a = 3, b = 3, c = 3, \) is obviously inconsistent. Note that \( G_{22} \) is value dependent on \( G_{21}. \) Although the effect of \( G_{11} \) is undone, its effect is already propagated to LS3 which is not undone. □

We shall now explore an alternative approach which prevents the partial effects of a global transaction from being read by other global transactions before its compensating subtransactions are executed.

Let \( AC(G) \) denote the set of data items that \( G \) accesses and commits, and let \( WC(G) \) denote the set of data items that \( G \) writes and commits. Suppose \( G_i \) is a compensated-for global transaction. Following the stipulation regarding the independence of \( CG_i, \) we see that any write operations of other global transactions can be executed between \( G_i \) and \( CG_i, \) as long as the local concurrency control criteria are followed. However, the read operations of global transactions must be carefully scheduled to ensure that the partial effects of a global transaction will not be read to other global transactions before it commits. A concurrency control correctness criterion, termed compensation serializability, is defined as follows:

**Definition 2 (Compensation serializability)** A global schedule \( S \) is compensation serializable if \( S \) is serializable and, for any global transaction \( G_j \) which is serialized between a compensated-for global transaction \( G_i \) and its compensating transaction \( CG_i \) in \( S, WC(G_i) \cap AC(G_j) = \emptyset. \)

Thus, in a compensation serializable global schedule, any partial effects of a compensated-for global transaction will remain unseen by other global transactions. As a result, each
global transaction always reads a consistent global database state. We have the following straightforward lemma:

**Lemma 1** Every global transaction in a compensation serializable global schedule reads a consistent multidatabase state.

Since the execution of a global subtransaction always results in a consistent local database state, a local transaction therefore always reads such a state. Thus, all local transactions in $S$ read consistent local database states, and all global transactions read consistent multidatabase states. We claim that a compensation serializable global schedule $S$ always results in a consistent global database state. This is stated and proven succinctly in the following theorem:

**Theorem 1** A global schedule $S$ that is compensation serializable preserves multidatabase consistency.

**Proof:** Since $S$ is serializable, we assume that $S$ is conflict equivalent to a serial schedule $S'$ [BHG87]. By the semantics of compensation, the partial effects of compensated-for subtransactions in $S'$ are semantically compensated by their compensating transactions. Since no effects of compensated-for subtransactions are seen by other global subtransactions before they are compensated, any inconsistencies caused by these compensated-for subtransactions before they are compensated are restored by their compensating transactions. Let $S''$ be $S'$ restricted to those transactions that are neither compensated-for subtransactions nor their compensating transactions. Thus, $S''$ consists only of atomic local and global transactions [BHG87]. If each transaction in $S''$ sees a consistent database state, then $S''$ preserves the multidatabase consistency. Since all local transactions or global subtransactions at each local site in $S''$ either commit or abort, every local transaction sees a consistent local database state. Following Lemma 1, every global transaction also sees a consistent multidatabase state. Hence, $S''$ preserves multidatabase consistency. 

5 Ensuring Compensation Serializability

In this section, we present a GTM scheduling protocol, which ensures compensating serializability on the execution of local and global transactions.
5.1 Execution of Compensating Subtransactions

We assume that the GTM submits global transactions to the LDBSs through servers, which are associated with the LDBSs and act as the interface between the GTM and the LDBSs. Each global subtransaction is then submitted to an individual LDBS by a server as a single transaction. The commit or abort of these submitted global subtransactions is acknowledged by the LDBSs to the GTM through these servers. Moreover, a data item which is used as a ticket by the GTM can be created at each local site for the purpose of multidatabase concurrency control. We also assume that the compensating subtransactions are submitted together with the corresponding compensatable subtransactions.

The GTM maintains state information for each subtransaction. This state is inactive if the subtransaction has not started its execution, is active if the subtransaction has started but not committed, is committed if the subtransaction has committed, is aborted if the subtransaction has aborted, and is committed-reversed if both the subtransaction and its compensating subtransaction have committed.

Each global subtransaction \( G_{ij} \) initially is in the “inactive” state. When it is submitted for execution, \( G_{ij} \) then enters the “active” state. If \( G_{ij} \) is committed in the local site, the server at the local site sends a commit acknowledgement \( <\text{ack\_commit}, G_{ij}> \) to the GTM; otherwise, if \( G_{ij} \) is aborted in the local site, the server at the local site sends an abort acknowledgement \( <\text{ack\_abort}, G_{ij}> \) to the GTM. When the GTM receives a commit acknowledgement for subtransaction \( G_{ij} \), it updates the state of \( G_{ij} \) to be committed. When the GTM receives an abort acknowledgement for subtransaction \( G_{ij} \), it updates the state of \( G_{ij} \) to be aborted.

When an \( <\text{ack\_abort}, G_{ij}> \) is received, the GTM aborts the global transaction \( G_i \) by aborting all its subtransactions that are in the active state and compensates for all subtransactions that are in the committed state. When a subtransaction must be compensated for, the compensating subtransaction may need to be retried until it actually succeeds. When the compensating subtransaction commits, the state of the original subtransaction in the GTM changes to committed-reversed.

We say that a global transaction is robustly terminated if all its subtransactions have been marked either as committed or as aborted or committed-reversed.
5.2 A Revised Transaction-Site Graph Approach

We now address the issues that arise in connection with the enforcement of compensation serializability, and with its essential prerequisite, the ensuring of serializability. Much research of both a theoretical and a practical nature has been directed to maintaining serializability in the MDBS environment [GRS91, VW92, MRB+92, ZE93]. Among the proposed approaches, the ticket method [GRS91] has been recognized to be of practical utility. With this method, each global transaction must update the ticket data item at the local site it accesses. Consequently, every global transaction conflicts with every other global transaction at each local site where both have subtransactions. Serializability on a global schedule can then be ensured if the local schedules are serializable and the serialization graph of its global subschedule is acyclic [GRS91]. Both conservative and non-conservative approaches have been proposed to maintain an acyclic serialization graph of a global subschedule. The conservative approaches, such as site graph [BS88] and serialization events [ED90, MRB+92, Pu88], can avoid a large number of transaction aborts but may provide a low degree of concurrency and involve a high overhead. The non-conservative approaches, such as optimistic ticket method [GRS91], provides a high degree of concurrency but subject to a high percentage of transaction aborts, which may be too expensive. In addition, as pointed out in [HHS93], the non-conservative approaches may severely degrade local transaction throughput and thus may be undesirable in the MDBS environment.

We propose a compromising approach, which effectively combines the ticket method, the conservative serialization graph testing [BH87], and the site graph to avoid the high overhead on maintaining serialization events, to prevent transaction aborts, and to make the approach fault-tolerant. The conservative serialization graph testing maintains a stored serialization graph (SSG) among global transactions for scheduling purposes. In order to describe the commitment status of global subtransactions at local sites, we modify the concept of SSG by adding site nodes and their incident edges, as follows:

**Definition 3 (Stored Transaction-site Graph)** The stored transaction-site graph of the execution of a set of global transactions in global schedule $S$, denoted $STG$, is a directed graph whose nodes are the global transactions (transaction nodes) and local sites (site nodes) and whose edges are all $G_i \rightarrow G_j (i \neq j)$ and $G_i \rightarrow LS_j$ such that

---

*Due to non-serializability and deadlocks.*
• $G_i \rightarrow G_j$ if an operation of $G_{ik}$ precedes and conflicts with an operation of $G_{jk}$, for $k = 1, \ldots, n$.
• $G_i \rightarrow LS_j$ if $G_i$ accesses $LS_j$.

If the GTM receives an $<ack\_commit, G_{ij}>$ message from $LS_j$, then edge $G_i \rightarrow LS_j$ is referred to as a committed edge. If the GTM receives an $<ack\_abort, G_{ij}>$ message from $LS_j$, then edge $G_i \rightarrow LS_j$ is referred to as an aborted edge. If the GTM has received no acknowledgement of a commit or abort of $G_{ij}$ from $LS_j$, then edge $G_i \rightarrow LS_j$ is referred to as a unmarked edge.

We assume that each global transaction predeclares its read operation set and write operation set. We say that a global transaction $G_j$ accesses undeterminedly from $G_i$ in an STG if (1) $G_i \rightarrow G_j$; (2) there is a local site $LS_k$ such that $w(x)$ and $op(x)$ are operations of $G_{ik}$ and $G_{jk}$, respectively, and $G_i \rightarrow LS_k$ is not an "aborted" edge; and (3) if there is some $G_l$ such that $G_i \rightarrow G_l$, $G_l \rightarrow G_j$, and $w(x)$ is an operation of $G_l$, then $G_l \rightarrow LS_k$ is an "aborted" edge.

Let $S$ be a global schedule and $G = \{G_1, \ldots, G_m\}$ be a set of global transactions in $S$. We denote $STG|_G$ as STG restricted only to transaction nodes in the STG. The GTM scheduling protocol includes an edge insertion rule, an edge deletion rule, and an operation submission rule. Edges incident to a transaction node $G_i$ are inserted into or deleted from the STG only if the rules below are followed:

Edge Insertion Rule: Insertion of $G_i \rightarrow LS_j$ for each local site $LS_j$ that $G_i$ accesses does not result in $G_{ij}$ accessing undeterminedly from any global transaction which is previously scheduled in the STG; and insertion of $G_k \rightarrow G_i$ for every previously scheduled $G_k$ in the STG that conflicts with $G_i$ does not result in a cycle in $STG|_G$.

Edge Deletion Rule: Edges incident on a global transaction are deleted from the STG as soon as the global transaction has robustly terminated and has no incoming edges in the STG.

The operations of a global transaction $G_i$ are submitted to LDBSs for execution only if the edges of the global transaction have been successfully inserted into the STG. The operations are submitted to servers based upon the following rule:
Operation submission rule: Each operation is submitted only after all conflicting operations of previously scheduled global transactions have been acknowledged.

Lemma 2 Consider two conflicting global transactions $G_i$ and $G_j$ in an STG. $G_i$ is serialized before $G_j$ in global schedule $S$ if and only if the edges of $G_i$ are inserted into STG before the edges of $G_j$ are inserted into the STG.

Proof: (if) We need to show that, if the edges of $G_i$ are inserted into the STG prior to the edges of $G_j$, then $G_i$ is serialized before $G_j$. Suppose $G_i$ is not serialized before $G_j$ in global schedule $S$. Since $G_i$ conflicts with $G_j$, there must be conflicting operations $op_i$ of $G_i$ and $op_j$ of $G_j$ such that $op_j$ is executed before $op_i$ in $S$. Hence, there is an edge $G_j \rightarrow G_i$ in the STG, which contradicts the assumption.

(only if) Conversely, we need to show that, if $G_i$ is serialized before $G_j$, then the edges of $G_i$ are inserted into the STG prior to the edges of $G_j$. Suppose the edges of $G_j$ are inserted into the STG before the edges of $G_i$. Since $G_i$ conflicts with $G_j$, there must be an edge $G_j \rightarrow G_i$ in the STG. By the operation submission rule, there are conflicting operations $op_i$ of $G_i$ and $op_j$ of $G_j$ such that $op_j$ is executed before $op_i$ in global schedule $S$. Hence, $G_j$ must be serialized before $G_i$ in $S$, which contradicts the assumption.

Theorem 2 If the submissions of global transactions follow the GTM scheduling protocol, then the serializability of local schedules implies the compensation serializability of global schedules.

Proof: Clearly, the edge insertion and deletion rules generate an STG which is more restrictive than the SSG. Thus, the global subschedule is serializable. In addition, since each global transaction conflicts with all other global transactions at each local site where they both have subtransactions, the serialization order of global transactions at all local sites is relatively synchronized. Following the discussion in [GRS91, MRB+92, ZE93], the global schedule is serializable.

We now show that, for every global transaction $G_j$ in global schedule $S$, if a compensated-for global transaction $G_i$ is serialized before $G_j$ in $S$ and $WC(G_i) \cap RC(G_j) \neq \emptyset$, then $CG_i$.

---

4 For any two global transactions $G_i$ and $G_j$ in $G$, the serialization orders of all global subtransactions of $G_i$ either precede or follow the serialization orders of all global subtransactions of $G_j$ at local sites.
is not serialized after $G_j$. Since $G_i$ conflicts with $G_j$ at a local site, for example, $LS_k$, and $G_i$ is serialized before $G_j$, then by lemma 2, $G_i \rightarrow LS_k$ must be inserted into the STG before $G_j \rightarrow LS_k$ is inserted. By the edge insertion rule, $G_j \rightarrow LS_k$ can be inserted into the STG only if $G_i \rightarrow LS_k$ is deleted from the STG, or it is an aborted edge. Since $G_i \rightarrow LS_k$ is a committed edge, due to the edge deletion rule, $CG_i$ must commit before the edges of $G_j$ are inserted into the STG. Since $CG_i$ conflicts with $G_j$, then by Lemma 2, $CG_i$ is not serialized after $G_j$. □

We have presented a method of graph testing which integrates serialization graph testing with the use of transaction-site graph. In contrast to the use of transaction-site graph proposed in [MRKS92], serialization graph testing has the potential of achieving higher concurrency in the execution of global transactions.

6 Concluding Remarks

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

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

References


