A Framework for Flexible Transaction Management in Multidatabase Systems

Aidong Zhang

Omran Bukhres

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

Report Number:
93-038

A Framework for Flexible Transaction Management in Multidatabase Systems

Aidong Zhang, Omran Bukhres
and
Ahmed K. Elmagarmid

CSD-TR-93-038
July 1993

A Framework for Flexible Transaction Management in Multidatabase Systems *

Aidong Zhang and Omran Bukhres and Ahmed K. Elmagarmid
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 USA

Abstract

Global transaction management and the preservation of local autonomy present conflicts to the design of multidatabase transaction management systems. A flexible transaction model for the specification of global transactions has been proposed to enhance global transaction management while preserving local autonomy. This paper presents a theory of flexible transaction management that is applicable in those situations where local database systems maintain only serializability and recoverability. A fundamental characterization of the model and of the properties of flexible transactions is first offered. The meaning of relaxed atomicity and isolation of flexible transactions is precisely defined. We then investigate the principles of flexible transaction management that are necessary for ensuring these properties. A class of flexible transactions, which can be executed in the presence of failures, is constructed, and a new correctness criterion is proposed. The results demonstrate that the flexible transaction model enhances substantially the scope of global transaction management beyond that offered by the traditional global transaction model.

1 Introduction

A multidatabase system (MDBS) serves to integrate a set of local database systems (LDBSs) at various local sites (LSs). The central concern of such an integration is the preservation of the local autonomy of the component database systems. Such aspects of autonomy as design, execution, and

*This work was supported by a Purdue Research Foundation Fellowship and a grant from the Software Engineering Research Center at Purdue University (a National Science Foundation Industry/University Cooperative Research Center — NSF Grant No. ECD-8913133).

control have been studied in [Lit86, GMK88, BS88, Pu88, DE89, Vei90, GRS91], and their impact on multidatabase transaction management is discussed in [DEK90].

MDBSs process two varieties of transactions. A local transaction accesses a local database only and is submitted directly to a local database system. A global transaction, on the other hand, may access several local databases. Such a global transaction is submitted to a global transaction manager (GTM), superimposed upon a set of local autonomous database systems, where it is parsed into a series of global subtransactions to be submitted to the local database systems. A global transaction management system serves to maintain the correct execution of global transactions, while each local database system maintains the correct execution of both local transactions and global subtransactions at its site. The obstacles to global transaction management in MDBSs arise primarily from the constraints posed by the autonomy of local database systems.

A flurry of research activity has been devoted to the problems of the concurrency control and atomic commitment of global transactions [Pu88, DE89, GRS91, MRKS92a, MRB$^+$92]. The results of these endeavors reveal the limitations of global transaction management when the traditional global transaction model is employed in MDBSs. To increase the global applicability of MDBSs, an extended transaction model, termed flexible transactions[1], has been proposed [RELL90, ELLR90] for the specification of global transactions. The fundamental characteristic of this model is its provision of alternative choices for the execution of subtransactions of global transactions. Consequently, the execution of global transactions becomes more resilient to failures, in that the aborting of individual subtransactions may not prevent the whole global transaction from "successfully" executing. A weaker concept of the atomicity of global transactions is thus permitted. The traditional concept of the isolation of global transactions is also relaxed by allowing a global transaction to reveal its partial effects of compensatable subtransactions to other global transactions prior to its commitment. Since this model was proposed, much research has been devoted to its application [LEB92, ARNS92, ANRS92, KPE92]. Most of this work has assumed the availability of prepare-to-commit states [BHG87] at local sites. In such a scenario, the management of flexible transactions is relatively straightforward.

In contrast to other well-defined extended transaction models, such as nested transactions [Mos81], multi-level transactions [BSW88], and sagas [GMS87], the concept of relaxed atomicity and isolation has only been vaguely defined in respect to flexible transactions. Flexible transaction management in the MDBS environment thereby lacks a theoretical basis. Moreover, many researchers have pointed out that some local database systems may not support prepare-to-commit

[1]Other extended transaction models, such as those appeared in [Eea92], are also proposed. Since they are not approached particularly for the MDBS environment. We will not discuss them further.

states. In these instances, a local database system that participates in a multidatabase environment may unilaterally abort a global subtransaction without agreement from the global level (termed a local unilateral abort). It may also be a violation of local autonomy to require such local database systems to provide prepare-to-commit states. Thus, the demands of local autonomy considerably increase the difficulty of ensuring that a single logical action (commit or abort) of a flexible transaction is consistently carried out at multiple local sites. In addition, even when the local database systems do provide such support, the potential blocking and long delays caused by using prepare-to-commit states would severely degrade local execution autonomy.

In this paper, we offer a precise definition of the fundamental model and of the properties of flexible transactions. We present a theory of flexible transaction management in the MDBS environment in which the local database systems are required only to ensure serializability and recoverability [BHG87]. In the proposed formulation, a flexible transaction is defined as a set of subtransactions upon which a set of partial orders is specified. Each partial order provides one alternative to the successful execution of the flexible transaction. This methodology differs from the previous approach in that no specific semantics of applications are involved. Therefore, a theoretical basis for flexible transaction management can be built. We then classify the set of flexible transactions that can be executed in an error-prone MDBS environment. As compensation and retry approaches are unified and employed as flexible transaction failure recovery techniques, local prepare-to-commit states are no longer required. A new correctness criterion for the concurrent execution of flexible transactions, termed compensating serializability, is also proposed, which prevents any inconsistent partial effects of a flexible transaction to be seen by other flexible transactions. The results to be presented demonstrate that the flexible transaction model enhances substantially the scope of global transaction management beyond that offered by the traditional global transaction model.

This paper is organized as follows. Section 2 introduces the fundamental model and properties of flexible transactions. In Section 3, we construct those flexible transactions that can be executed in the error-prone MDBS environment without requiring local prepare-to-commit states. In Section 4, we discuss the effect of compensation on concurrency control of flexible transactions and propose a new correctness criterion. Concluding remarks are offered in Section 5.

2 The System of Flexible Transactions

In this section, we precisely define the fundamental flexible transaction model that specifies global transactions. This model is based upon the initial work proposed in [RELL90]. We then discuss the properties of such flexible transactions.

2.1 Definitions

Following [BHG87, Had88], 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. For the elements of a transaction, we denote the four basic operations as follows: $r(x), w(x), c$, and a (possibly subscripted), where $r(x)$ and $w(x)$ are *read* and *write* operations, and c and a are *commit* and *abort* termination operations. We 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. Two operations *conflict* with each other if they access the same data item and at least one of them is a *write* operation. A local transaction is a transaction that is submitted directly to an LDBS. A global transaction, that is submitted to the GTM, is defined as a set of subtransactions where each subtransaction is a transaction accessing the data items at a single local site.

The concept of *flexible transactions* offers a more flexible extension of the above traditional global transaction model. Each global application may be accomplished by a repertoire of alternative subtransactions, only a subset of which must be successful. The definition of flexible transactions takes the form of a high-level applications description. Various applications semantics, such as commit dependencies, abort dependencies, and the acceptable set of successful subtransactions, are captured in the flexible transaction definition. Such a semantic-oriented formulation of flexible transactions may not prevent redundancy in the dependency specification, and the structure of flexible transactions cannot generally be depicted. Drawing a generic structure of flexible transactions is thus necessary for the discussion of flexible transaction management.

In this section, we formalize the fundamental flexible transaction model. The strategy to be used is similar to that of other extended transaction models, such as nested transactions [Mos81], multi-level transactions [BSW88], and sagas [GMS87]. We basically define each flexible transaction as a set of subtransactions upon which a set of partial orders is specified. Each partial order of subtransactions defines a possible execution of the flexible transaction.

Let $T = \{t_1, t_2, ..., t_n\}$ be a set of subtransactions. An *ordering relation* $\prec_p$ on a subset T^p of T defines an irreflexive transitive relation on T^p, with at most one subtransaction at each local site in T^p[2]. The pair $(T^p, \prec_p)$ is a *partial order* of subtransactions. The ordering relation $\prec_p$ specifies the precedence and simultaneity of the execution of the subtransactions in T^p. If $t_i \prec_p t_j$, then t_i must succeed before t_j is executed; otherwise, t_i and t_j can execute simultaneously. A subtransaction may involve more than one ordering relation. The chosen priority of these ordering relations must

[2]This is necessary since serializability will be used for the concurrency control of flexible transactions [GPZ86].

be specified. For instance, if $t_i \prec_{p_1} t_j$ and $t_i \prec_{p_2} t_k$ and $\prec_{p_1}$ has higher priority than $\prec_{p_2}$, then t_k will be executed only when t_j fails. Given a set P of ordering relations defined on the subsets of T, we rank the chosen priority of these ordering relations by giving subscripts to the elements of P. That is, if $P = \{\prec_{p_1}, ..., \prec_{p_h}\}$, then, for $1 < i < j < h$, the subtransactions that are specified by p_i have higher priority to be chosen for execution than the subtransactions that are specified by p_j. We call such ordering relations *ranked ordering relations.*

We now formally define flexible transactions as follows.

Definition 1 (Flexible transaction) *A flexible transaction $G = (T, P)$ consists of a set T of subtransactions and a set P of ranked ordering relations, where each ordering relation in P is defined on a subset of T (forming a partial order). The successful execution of G is indicated by the commitment of all and only the subtransactions in one partial order of G.*

Semantically, an ordering relation specifies the execution dependencies (or control flow) among the subtransactions of a flexible transaction. Correct parallel (simultaneity) and sequential (precedence) execution among the subtransactions in each partial order of a flexible transaction are therefore specified. In addition, multiple ordering relations to be defined on a flexible transaction provide alternative control flows for the execution of the flexible transaction. To make the structure of a flexible transaction more visible, we also describe such execution dependencies of a flexible transaction by a graph as follows:

Definition 2 (Execution dependency graph) *An execution dependency graph of flexible transaction $G = (T, P)$, denoted $EDG(G)$, is a directed graph whose nodes are all subtransactions of G and whose edges are all $t_i \xrightarrow{\rho} t_j$ ($t_i, t_j \in T$), where $\rho \subseteq P$ is a set of ordering relations*[3] *such that t_i precedes t_j in an ordering relation $\prec_p \in \rho$ and there is no other subtransaction t_k such that it follows t_i and precedes t_j in $\prec_p$.*

We now consider an example which is given in [ELLR90].

Example 1 *Consider a travel agent information system. A global transaction G_1 in this system may consist of the following subtasks:*

- *Customer calls the agent to schedule a trip.*
- *Agent negotiates with airlines for flight tickets.*
- *Agent negotiates with car rental companies for car reservations.*

[3]For simplicity, we use p to refer to $\prec_p$ in the execution dependency graph.

- *Agent negotiates with hotels to reserve rooms.*
- *Agent receives tickets and reservations and then gives them to the customer.*

Let us assume that, for the purpose of this trip, the only applicable airlines are Northwest and United, the only car rental company is Hertz, and three hotels in the destination city are Hilton, Sheraton and Ramada. The travel agent can then order a ticket from either Northwest or United airlines. Similarly, the agent can reserve a room for a customer at any of the three hotels. Based on these observations, the travel agent may choose from the following subtransactions:

t_1: *Order a ticket from Northwest Airlines;*

t_2: *Order a ticket from United Airlines, if t_1 fails;*

t_3: *Rent a car from Hertz;*

t_4: *Reserve a room from Hilton;*

t_5: *Reserve a room from Sheraton, if t_4 fails;*

t_6: *Reserve a room from Ramada, if t_4 and t_5 fail.*

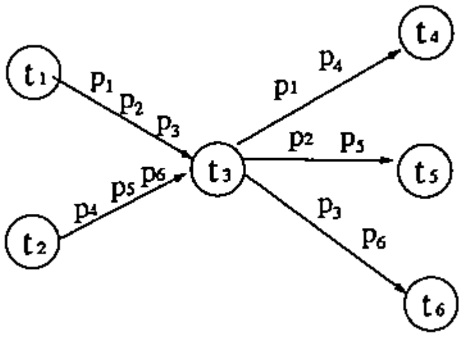
In this example, t_1 and t_2 are two alternative subtransactions for ordering a ticket. In this case, t_2 will be executed if subtransaction t_1 fails to achieve its objective. Similarly, t_4, t_5 and t_6 are alternative subtransactions for reserving a room.

There are six ordering relations $\prec_{p_1}, \prec_{p_2}, \prec_{p_3}, \prec_{p_4}, \prec_{p_5}$, and $\prec_{p_6}$ that are defined on the subsets of $\{t_1, t_2, t_3, t_4, t_5, t_6\}$:

$\{t_1 \prec_{p_1} t_3, t_3 \prec_{p_1} t_4, t_1 \prec_{p_1} t_4\}$, $\{t_1 \prec_{p_2} t_3, t_3 \prec_{p_2} t_5, t_1 \prec_{p_2} t_5\}$, $\{t_1 \prec_{p_3} t_3, t_3 \prec_{p_3} t_6, t_1 \prec_{p_3} t_6\}$,

$\{t_2 \prec_{p_4} t_3, t_3 \prec_{p_4} t_4, t_2 \prec_{p_4} t_4\}$, $\{t_2 \prec_{p_5} t_3, t_3 \prec_{p_5} t_5, t_2 \prec_{p_5} t_5\}$, $\{t_2 \prec_{p_6} t_3, t_3 \prec_{p_6} t_6, t_2 \prec_{p_6} t_6\}$.

In $EDG(G_1)$, we have:



□

Each subtransaction of a flexible transaction is either compensatable or non-compensatable. A subtransaction is *compensatable* if the effects of its execution at a local site can be semantically undone, after it commits, by executing a compensating transaction. Compensatable subtransactions play an important role in the flexible transaction management. A partial execution of a partial order of subtransactions can be discarded only when the effect of its committed subtransactions can be undone.

In each partial order of subtransactions, the data dependencies among operations in different subtransactions define data flow among the subtransactions. Let flexible transaction G have subtransactions $t_1, t_2, \cdots, t_n$. We say that t_{j_t} is *data-dependent* on $t_{j_1}, ..., t_{j_{t-1}}$ $(1 \leq j_1, ..., j_t \leq n)$, denoted $t_{j_1} \rightarrow_d t_{j_t}$, $t_{j_2} \rightarrow_d t_{j_t}$, ..., $t_{j_{t-1}} \rightarrow_d t_{j_t}$, if the execution of one or more operations in t_{j_t} is semantically determined by the values read by $t_{j_1}, ..., t_{j_{t-1}}$.

We have formulated two types of dependencies among the subtransactions of a flexible transaction: execution and data dependencies. In the remainder of this paper, we assume that these dependencies are the only relationships in effect among the subtransactions of each flexible transaction.

2.2 Properties

In a manner similar to traditional transactions, a flexible transaction must be a unit of consistent and reliable computation. Thus, we must provide the means to justify the consistency and reliability of the execution of a flexible transaction. Traditionally, the ACID properties (atomicity, consistency, isolation, and durability) [Gra81, HR83, ÖV91] have been advanced as the justification of the consistency and reliability of transactions. While some of these properties are applicable to flexible transactions, others are not. Clearly, the concept of atomicity must be relaxed for flexible transactions, since some subtransactions in one partial order of a flexible transaction may be aborted while the flexible transaction as a whole succeeds. We formulate below the fundamental properties of flexible transactions that are necessary and sufficient for the justification of the consistency and reliability of flexible transactions.

We first discuss the weaker concept of atomicity for flexible transactions. Although the traditional understanding of atomicity may no longer be required for flexible transactions, a certain degree of atomicity must still be ensured to produce correct executions. That is, either all and only the subtransactions in one partial order of a flexible transaction commit or none of the subtransactions of this flexible transaction does. Using a compensation approach, the compensatable subtransactions in multiple partial orders of a flexible transaction can be executed and committed simultaneously, as long as any partial effects of the flexible transaction will eventually be compensated. Combining the semantics of compensation with the above weaker concept of atomicity of flexible transactions, we define the semi-atomicity of flexible transactions as follows:

Property 1 (Semi-atomicity) *A flexible transaction is semi-atomic if either all and only the effects of its subtransactions in one partial order or no partial effects of its subtransactions are made permanent in local databases.*

The traditional consistency property is inherited by flexible transactions. 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. In a multidatabase system, there are two types of integrity constraints: local integrity constraints are defined on data items in a single local site, while global integrity constraints are defined on data items in multiple local sites. A local transaction or a subtransaction of a flexible transaction is *locally consistent* if it preserves local integrity constraints. As defined for traditional global transactions, the execution of a flexible transaction as a single unit should map one consistent global database state to another. Thus, a flexible transaction must preserve both local and global integrity constraints. However, to be different from the traditional global transactions, this consistency of flexible transactions actually has to require that the execution of each partial order of subtransactions must map one consistent global database state to another. We give the consistency property of flexible transactions as follows:

Property 2 (Global consistency) *A flexible transaction is globally consistent if the execution of every partial order of subtransactions transfers the global database from one consistent state to another.*

To this point, the concept of relaxed isolation has only been vaguely defined in respect to flexible transactions. To achieve high concurrency on the execution of flexible transactions, it was proposed in [ELLR90, Leu91, LEB92] to release the results of compensatable subtransactions of a flexible transaction prior to the commitment of the flexible transaction. The issue is whether a flexible transaction can see the intermediate results of another flexible transaction while both are executing. In contrast to sagas [GMS87], a flexible transaction does not require that any of its subtransactions alone preserve global consistency. Thus, any intermediate results of a flexible transaction may be globally inconsistent. However, the results of compensatable subtransactions of a flexible transaction which do preserve global consistency may be seen by other flexible transactions before the flexible transaction commits. Such revealed partial results may eventually have to be compensated. Section 4 contains more detailed discussion on this issue.

Property 3 (Flex-isolation) *A flexible transaction is flex-isolated if it can reveal only its globally consistent partial results to other flexible transactions.*

The durability of flexible transactions may be defined as similar to the traditional concept. For completeness, we provide it as follows:

Property 4 (Durability) *A flexible transaction is durable if, despite failures, the results of all its committed subtransactions are made permanent in the database.*

We say that a flexible transaction management scheme is correct if it guarantees the execution of flexible transactions satisfying Properties 1-4.

As usual, the global consistency of flexible transactions is ensured by the writers of those transactions, while the durability of subtransactions is ensured by LDBSs. The durability of flexible transactions is therefore ensured. It is the responsibility of the GTM to ensure the semi-atomicity and flex-isolation of flexible transactions. Since any intermediate results of a flexible transaction are generally not guaranteed to be globally consistent, we use serializability[4] as the basic concurrency control correctness criterion for the execution of flexible and local transactions. In the following two sections, we shall investigate additional conditions that are enforced on flexible transactions and their execution to preserve the semi-atomicity and flex-isolation.

3 Constructing Flexible Transactions for Ensuring Semi-atomicity

In this section, we focus on the preservation of the semi-atomicity of flexible transactions. We will formulate those conditions on flexible transactions that are sufficient for them to be correctly executed in an error-prone MDBS environment without requiring local prepare-to-commit states. The results to be presented show that the flexible transaction model allows the GTM to run more applications than does the traditional global transaction model.

3.1 Well-structured Flexible Transactions

We first introduce related concepts and then discuss the requirements on the execution dependencies of a flexible transaction that are necessary to preserve its semi-atomicity.

The semi-atomicity of a flexible transaction requires that all and only those subtransactions in one of its partial orders commit. As local prepare-to-commit states are not pre-assumed in

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

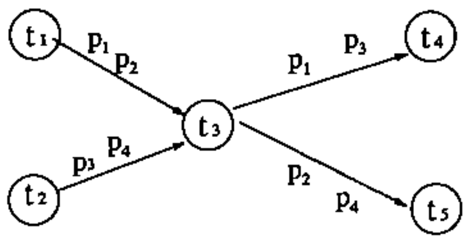
our scenario, a local database system may unilaterally abort a subtransaction without agreement from the global level. As a result, it becomes difficult to ensure that a single logical commit action of the subtransactions in one partial order of a flexible transaction is consistently carried out at multiple local sites. To handle local unilateral aborts while ensuring the semi-atomicity of a flexible transaction, the GTM may either re-execute its aborted subtransactions until they commit (forward approach) [BST90, WV90, MRKS92a] or undo the effects of its committed subtransactions (backward approach) [GM83, GMS87, LKS91a]. Approaches using forward recovery (redo and retry) and backward recovery (compensation) have been proposed in the literature to address the issue of preserving the semantic atomicity [GM83] of global transactions in MDBSs. In our scenario, we focus our investigations upon a unification of the retry and compensation approaches for the commitment of flexible transactions.

We further classify the non-compensatable subtransactions into two categories: retriable and pivot subtransactions. A subtransaction is *retriable* if it can commit after a finite number of resubmissions. A subtransaction is *pivot* if it is neither retriable nor compensatable.

In [MRKS92a], a basic multidatabase transaction model is proposed for the scenario in which local database systems do not support prepare-to-commit states. This model formulates each global transaction as the combination of a set of compensatable subtransactions, a set of retriable subtransactions, and a single pivot subtransaction. Any of these three parts of a global transaction is optional. Also, the subtransactions must not have any dependencies among them. 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 now explore the extension of the above model to flexible transactions. Let flexible transaction $G = (T, P)$ have subtransactions $t_1, t_2, ..., t_n$. Clearly, if $t_i \prec_p t_j$ in $\prec_p$ of P, then the commitment of t_i must precede that of t_j. We say that a partial order $(T^p, \prec_p)$ of G is *primitive* if T^p includes at most one pivot subtransaction; for any subtransaction t_i preceding the pivot subtransaction in $\prec_p$, t_i is a compensatable subtransaction; and for any subtransaction t_i following the pivot subtransaction in $\prec_p$, t_i is a retriable subtransaction. Using compensation and retry approaches similarly to [MRKS92a], the semantic atomicity of a primitive partial order of subtransactions can be preserved. However, to preserve the semi-atomicity of a flexible transaction, it is not necessary to require that all partial orders of a flexible transaction be primitive. The following example is illustrative:

Example 2 *Assume that a flexible transaction $G_1 = (T_1, P_1)$ is defined by the following execution dependency graph:*



Suppose that t_1 and t_2 are compensatable and t_3 is pivot. If t_4 is either compensatable or pivot, then $(T_1^{p_1}, \prec_{p_1})$ and $(T_1^{p_3}, \prec_{p_3})$ are not primitive. If t_1 and t_3 have already committed, and then t_4 aborts, the partial effects of $(T_1^{p_1}, \prec_{p_1})$ cannot be undone. However, the execution of t_4 can be replaced by the execution of t_5. As long as t_5 is retriable, G_1 can be committed. □

We now formulate those conditions on the execution dependencies of a flexible transaction which are necessary for preserving its semi-atomicity.

Let $G = (T, P)$ be a flexible transaction and $(T^p, \prec_p)$ be a partial order of G. We select a pivot subtransaction in T^p as the *principal pivot subtransaction* of T^p if it is a pivot subtransaction in T^p such that no other pivot or retriable subtransaction precedes it in $\prec_p$. A subtransaction t_i in T^p is *abnormal* if one of the following conditions is satisfied:

- t_i is a compensatable subtransaction and there is a pivot or retriable subtransaction t_j in T^p such that $t_j \prec_p t_i$; or
- t_i is a pivot subtransaction, but not a principal one.

Otherwise, t_i is a *normal* subtransaction. Obviously, all subtransactions in a primitive partial order are normal. A principal pivot subtransaction is also normal. Following the above definition, we see that only a compensatable or pivot subtransaction may be an abnormal subtransaction. Let $(T^p, \prec_p)$ be a partial order of subtransactions that is not primitive. When a pivot or retriable subtransaction commits, the effect of $(T^p, \prec_p)$ in the database can no longer be undone. Thus, in case any abnormal subtransaction $t_i \in T^p$ aborts, appropriate actions must be sought to continue the execution of G. By utilizing the flexibility of flexible transactions, if there is an alternative subtransaction whose execution can semantically replace the execution of t_i, then the semi-atomicity of G may still be preservable. To formalize such flexible transactions, we define *well-structured* flexible transactions as follows:

Definition 3 (Well-structured flexible transaction) *A flexible transaction $G = (T, P)$ is well-structured if, for each abnormal subtransaction t_i participating in $(T^{p_m}, \prec_{p_m})$ ($\prec_{p_m} \in P$), there is* *an alternative subtransaction t_j participating in $(T^{p_n}, \prec_{p_n})$ ($\prec_{p_n} \in P$) such that the aborting of t_i will lead the execution of G to t_j without resulting in any database inconsistency.*

Following Definition 3, for any abnormal subtransaction in a partial order $(T^{p_m}, \prec_{p_m})$ of a well-structured flexible transaction, there is one or more alternative subtransactions in another partial order $(T^{p_n}, \prec_{p_n})$ that perform the equivalent function. This also implies that, in a flexible transaction, any partial order such that there is no alternative subtransaction for its subtransactions must be primitive. Hence, there must be at least one primitive partial order in a well-structured flexible transaction. For instance, in Example 2, to ensure G_1 is well-structured, t_4 can be compensatable, pivot, or retriable. However, t_5 must be retriable. Hence, $(T_1^{p_1}, \prec_{p_1})$, $(T_1^{p_2}, \prec_{p_2})$, and $(T_1^{p_3}, \prec_{p_3})$ may not be primitive, but $(T_1^{p_4}, \prec_{p_4})$ must be primitive.

Because a partial order of subtransactions that is not primitive is permitted, a well-structured flexible transaction extends the scope of global transactions that can be specified in the MDBS environment in contrast to the basic multidatabase transaction model proposed in [MRKS92a].

3.2 Recoverable Flexible Transactions

We now discuss the additional conditions on data dependencies of well-structured flexible transactions which are necessary for preserving their semi-atomicity.

Following [BHG87, Had88], we define a *schedule over a set of transactions* as a partial order of the operations of those transactions which orders all conflicting operations and which respects the order of operations specified by the transactions. A local schedule S_k is a schedule over both local transactions and global subtransactions which are executed at local site LS_k. A global schedule S is a schedule over both local and global transactions which are executed in an MDBS. We denote $o_1 <_S o_2$ if operation o_1 is executed before operation o_2 in schedule S. In the following discussion, we assume that all global transactions in $\mathcal{G}$ are well-structured flexible transactions.

We have shown that, when subtransactions in a partial order of a flexible transaction can be executed in parallel, the types of subtransactions are used to determine their commitment order for preserving the semi-atomicity. For those subtransactions which are retriable, we also observe here that data dependencies must be considered in determining their commitment order. By definition, the retriability of a subtransaction is purely determined by its semantics. In [MRKS92a], because there are no dependencies between subtransactions, the retrial of a subtransaction has no effect on the execution of other subtransactions. However, in our context, the retrial of a subtransaction may also have effect on the commitment order of other subtransactions because of data dependencies among the subtransactions. For instance, let us assume that $t_1 \rightarrow_d t_2$ and t_1 is retriable. Suppose that t_2 commits, and t_1 aborts and then it is retried. A local transaction may be executed after t_1 is aborted but before it is retried at its local site, which may result in inconsistencies between the data read from the original execution of t_1 and from its retrial. As a result, an inconsistent database state may occur.

Let $G = (T, P)$ be a well-structured flexible transaction and t_i in T be a retriable subtransaction. To ensure that the retrial of t_i does not result in any database inconsistency, when a subtransaction t_j is data-dependent on t_i, the commitment of t_i must precede that of t_j. Thus, if the retrial of t_i leads to a result which is different from that of its original execution, then t_j that has read the data from the original execution of t_i may be aborted and re-executed. Consequently, each retriable subtransaction remains retriable without resulting in any database inconsistency as long as all other subtransactions that are data-dependent upon it have not committed. We formulate below the concept of *commit dependency* that is defined on two subtransactions in a partial order, incorporating all effects of data dependencies, execution dependencies, and the types of subtransactions on the commitment ordering of the subtransactions.

Let t_i and t_j be two subtransactions in a partial order $(T^p, \prec_p)$. We say that t_j is *commit-dependent* on t_i, denoted $t_i \rightarrow_c t_j$, if one of the following conditions is satisfied:

- $t_i \prec_p t_j$;
- $t_i \rightarrow_d t_j$ and t_i is retriable;
- t_i is normal and compensatable, and t_j is either pivot or retriable; or
- t_i is normal and pivot, and t_j is either pivot or retriable.

Clearly, to preserve the semi-atomicity of a flexible transaction by using compensation and retry approaches, the commitment order of the subtransactions in a partial order of a flexible transaction should follow their commit dependencies. We formulate this property in global schedules as follows:

Definition 4 (Intra-recoverability) *Let $\mathcal{G}$ be a set of well-structured flexible transactions. A global schedule S is intra-recoverable if, for each flexible transaction G in $\mathcal{G}$, and any two subtransactions t_i and t_j of G in S such that $t_i \rightarrow_c t_j$, $c_{t_j} \in S$ implies $c_{t_i} <_S c_{t_j}$.*

Following Definition 4, if a global schedule S is intra-recoverable, then each subtransaction in S can only commit after all subtransactions upon which it is commit-dependent have committed. Thus, if a normal and non-retriable subtransaction aborts, then only compensatable subtransactions may have been committed. These partial effects are therefore compensatable. If a retriable or abnormal subtransaction aborts, then the execution of the flexible transaction can either proceed by retrying the aborted subtransaction or by switching to the execution of an alternative subtransaction. Hence, the semi-atomicity of flexible transactions in $\mathcal{G}$ is always preservable.

The maintenance of the intra-recoverability of global schedules at the global level is determined by the characteristics of commit dependencies defined on the subtransactions in each partial order. Such dependencies can be described by a graph as follows:

Definition 5 (Commit dependency graph) *A commit dependency graph of a partial order $(T^p, \prec_p)$ of a flexible transaction $G = (T, P)$, denoted CDG$(T^p, \prec_p)$, is a directed graph whose nodes are all subtransactions of T^p and whose edges are all $t_i \rightarrow t_j$ $(t_i, t_j \in T^p)$ such that $t_i \rightarrow_c t_j$.*

The acyclicity of commit dependency graphs of partial orders provides a sufficient condition for maintaining global schedules as intra-recoverable. More precisely, we have the following theorem:

Theorem 1 *Let $\mathcal{G}$ be a set of well-structured flexible transactions. If, for each $G = (T, P)$ in $\mathcal{G}$, CDG$(T^p, \prec_p)$ is acyclic for all $\prec_p$ in P, then the intra-recoverability of global schedules can be ensured.*

Proof: Assume that, for each $G = (T, P)$ in $\mathcal{G}$, CDG$(T^p, \prec_p)$ is acyclic for all $\prec_p$ in P. Then, for any $G_i = (T_i, P_i)$ in $\mathcal{G}$ and $\prec_p \in P_i$, CDG$(T_i^p, \prec_p)$ may be topologically sorted. Without loss of generality, let $t_1, ..., t_m$ be the nodes of CDG$(T_i^p, \prec_p)$ and $j_1, ..., j_m$ be a permutation of 1,2,...,m such that $t_{j_1}, t_{j_2}, ..., t_{j_m}$ is a topological sort of CDG$(T_i^p, \prec_p)$. This order ensures that the commitment orders of these subtransactions in a global schedule conform to the definition of intra-recoverability. To illustrate this, let t_l and t_k be subtransactions in T_i^p such that $t_k \rightarrow_c t_l$. By the definition of CDG$(T_i^p, \prec_p)$, $t_k \rightarrow t_l$ is an edge in CDG$(T_i^p, \prec_p)$. Thus, t_k must appear before t_l in the topological sort $t_{j_1}, t_{j_2}, ..., t_{j_m}$. If the commitment order of all subtransactions in T_i^p follows the order of $t_{j_1}, t_{j_2}, ..., t_{j_m}$ in global schedule S, then the commitment of t_k precedes that of t_l in S. Hence, S is intra-recoverable. □

We define *recoverable* flexible transactions as follows:

Definition 6 (Recoverable flexible transaction) *A well-structured flexible transaction $G = (T, P)$ is recoverable if, for all $\prec_p$ in P, CDG$(T^p, \prec_p)$ is acyclic.*

Thus, if all flexible transactions are recoverable, then the intra-recoverability of global schedules can be ensured. Consequently, the semi-atomicity of the flexible transactions is preservable.

Clearly, all global transactions that follow the basic multidatabase model [MRKS92a] are also recoverable flexible transactions. In addition, the recoverable flexible transactions permit alternative execution dependencies and data dependencies to be defined in flexible transactions. The scope of global transactions that can be specified in the MDBS environment is therefore extended.

The retrial of the retriable subtransactions may also render unavoidable the non-serializable execution of flexible transactions, an unacceptable situation when serializability is required for the execution of flexible transactions. For instance, let flexible transaction G_1 have retriable subtransactions $t_1 : w(a)$ and $t_2 : w(c)$ at LS$_1$ and LS$_2$ respectively, and flexible transaction G_2 have retriable subtransactions $t_3 : w(a)$ and $t_4 : w(c)$ at LS$_1$ and LS$_2$ respectively. The following global schedule is then serializable:

$S : w_{t_1}(a)w_{t_3}(a)w_{t_2}(c)w_{t_4}(c).$

Suppose that t_1 and t_4 successfully commit, but t_2 and t_3 are aborted before c_{t_2} and c_{t_3} are executed due to failures at local sites LS_1 and LS_2. At this point, the global schedule becomes:

$S' : w_{t_1}(a)w_{t_4}(c)c_{t_1}c_{t_4}.$

The subtransactions t_2 and t_3 cannot be re-executed without causing the execution of flexible transactions G_1 and G_2 to be non-serializable. This difficulty may be solved by maintaining the commitment order of subtransactions at each local site as identical to their serialization order. We formulate this property in the global schedule as follows:

Definition 7 (Inter-recoverability) *A global schedule S is inter-recoverable if, for any two subtransactions t_i and t_j of different flexible transactions of $\mathcal{G}$ at local site LS_k, t_i is serialized before t_j and $c_{t_j} \in S$ implies $c_{t_i} <_S c_{t_j}$.*

Based upon the above discussion, a commitment protocol that maintains the intra-recoverability and inter-recoverability of global schedules can be designed at the global level. Such a protocol would control the submission of commit operations of subtransactions consistent with their commit dependencies and serialization orders. Such control of the commitment order of global subtransactions will not conflict with local recoverability, which is pre-assumed in our scenario. Consider t_1 and t_2 be two subtransactions at local site LS_k, with t_1 reading data item a from t_2 [BHG87]. There then exist $w_1(a) \in t_1$ and $r_2(a) \in t_2$ such that $w_1(a) <_S r_2(a)$. t_1 must then be serialized before t_2. The GTM must therefore control $c_{t_1} <_S c_{t_2}$ to maintain the inter-recoverability of S. At local site LS_k, following local recoverability, t_1 reading from t_2 [BHG87] implies $c_{t_1} <_{S_k} c_{t_2}$. A detailed discussion of such a protocol is beyond this paper and is not presented here.

4 Correctness of Global Schedules

In this section, we illustrate the necessity of preventing other flexible transactions from seeing the partial effects of a flexible transaction that is not guaranteed to be globally consistent. We also formulate a new correctness criterion for the concurrency control of flexible transactions. The effect of compensation on serializability is carefully analyzed. We assume here that all flexible transactions are recoverable.

We say that a subtransaction in global schedule S is *compensated-for* if it has committed in S and its effects need to be compensated. A flexible or global transaction G_i in global schedule S is *compensated-for* if it has compensated-for subtransactions in S. Thus, a compensated-for flexible transaction has some partial effects in local databases. However, it may already have committed all of the subtransactions in one of the partial orders of G_i.

Similarly to [MRKS92a], we consider a compensating transaction CG_i for flexible transaction G_i as a separate global transaction from G_i. CG_i consists of compensating subtransactions that compensate the compensated-for subtransactions of G_i to restore the database consistency. Moreover, CG_i should always be serialized after G_i in global schedules. Each compensating subtransaction must be retriable, since it does not make any sense to abort it [KLS90]. Each compensating subtransaction Ct_i for a compensatable subtransaction t_i must also be independent of the transactions that execute between t_i and Ct_i [MRKS92a]. Such independence is not required in the traditional concept of compensating transactions, as no uncontrolled interleaving of local transactions in the execution of global transactions occurs in that context. Local autonomy here requires that arbitrary local transactions must be executable while the compensating actions for a compensated-for flexible transaction are processed.

Following from Section 3, when a pivot or retriable subtransaction of a flexible transaction commits, all subtransactions in the partial order of the flexible transaction that include this committed subtransaction will commit. Consequently, the effect of this committed subtransaction must be part of the globally consistent state, and the effect can be seen immediately by other local or flexible transactions. However, when a compensatable subtransaction commits, it is not certain whether it will need to be compensated. If it does, then the results of this subtransaction are part of the partial effects that may not be globally consistent. Clearly, local transactions can see such partial effects of a compensated-for flexible transaction because the execution of a subtransaction always preserves local database consistency. The question now is whether other flexible transactions can see such partial effects of a compensated-for flexible transaction.

In [KLS90], a formal discussion is provided to analyze the situations in which a transaction may see the partial effect of another transaction before these partial effects are compensated. It is then generally elaborated in [LKS91a, LKS91b] that a global transaction should not be affected by both aborted and successful subtransactions of another global transaction. Otherwise, an inconsistent database state may be seen. A concurrency control correctness criterion, termed ***serializability with*** *respect to compensation (SRC)*, is further proposed in [MRKS92a] to preserve database consistency with the execution of global transactions which have no any type of dependencies among subtransactions in the MDBS environment. This criterion prohibits any global transaction that is serialized between a compensated-for global transaction G_i and its compensating global transaction CG_i to read from the local sites at which G_i aborts. However, this criterion is not applicable to a situation in which there are data-dependencies defined on global transactions [MRKS92b]. The following example is illustrative:

Example 3 *Consider an MDBS consisting of three LDBSs on D_1, D_2, and D_3, where data item a is in D_1, data item b is in D_2, and data item c is in D_3. Let the integrity constraints be $a < c$, $b < c$, and $a = b$. Let a global transaction G_1 consist of two subtransactions:*

$t_1 : r(a)w(a, a-1)$,

$t_2 : r(b)w(b, b-1)$.

Let another global transaction G_2 be:

$t_3 : r(a)$,

$t_4 : w(c, a+1)$.

Consider an execution of G_1 that results from database state $a = 3, b = 3, c = 5$, where t_1 commits and t_2 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 t_1. G_1, G_2, and CG_1 are serializable in the order $G_1 \rightarrow G_2 \rightarrow CG_1$. G_2 does not see any effect from 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 t_4 is data-dependent on t_3. □

Because a traditional global transaction is a special case of a flexible transaction, following Example 3, we see that even though flexible, compensated-for flexible, and compensating transactions are serializable and only the committed portion of a compensated-for flexible transaction is seen by other flexible transactions, global database consistency might not be retained. Thus, the partial effects of a compensated-for flexible transaction which is not guaranteed to be globally consistent should not be seen by other flexible transactions before its compensating transaction is executed. The results of such a subtransaction should be held from being seen by other flexible transactions until its effect is compensated.

Thus, we have clarified that releasing arbitrarily the effects of compensatable subtransactions prior to the commitment of the flexible transaction may not be appropriate. To permit as much concurrency as possible on the execution of flexible transactions and their compensating transactions, the flex-isolation of flexible transactions permits the effects of those compensatable subtransactions which are globally consistent to be seen by other flexible transactions before they are compensated. Let $RC(G)$ denote the set of data items that G reads and commits, and let $WC(G)$ denote the set of data items that G writes and commits. Let $G_i^{\natural}$ denote G_i restricted to the compensated-for subtransactions which do not guarantee global consistency. A concurrency control correctness criterion, termed *compensating serializability*, is defined as follows:

Definition 8 (Compensating serializability) *A global schedule S is compensating serializable if S is serializable and, for any flexible transaction G_j which is serialized between a compensated-for flexible transaction G_i and its compensating transaction CG_i in S, $WC(G_i^{\natural}) \cap RC(G_j) = \emptyset$.*

Thus, in a compensating serializable global schedule, any partial effects of a compensated-for flexible transaction that are not globally consistent will not be seen by other flexible transactions. As a result, each flexible transaction always sees a consistent global database state. We have the following straightforward lemma:

Lemma 1 *Every flexible transaction in a compensating serializable global schedule sees a consistent global database state.*

Since a subtransaction of a flexible transaction is also treated as a local transaction at a local site, its execution always results in a consistent local database state. Therefore, a local transaction always sees a consistent database state. Thus, all transactions in S see consistent database states. We claim that a compensating serializable global schedule S always results in a consistent global database state. This is stated and proved succinctly in the following theorem:

Theorem 2 *A global schedule S that is compensating serializable preserves global database 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 subtransactions and any inconsistency caused by these compensated-for subtransactions are restored. Let S'' be S' restricted to the transactions that are neither compensated-for subtransactions nor their compensating subtransactions. Thus, S'' consists of only traditional atomic local transactions [BHG87] and semi-atomic flexible transactions, if each transaction in S'' sees a consistent database state, then S'' preserves the global database 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 flexible transaction also sees a consistent global database state. Hence, S'' preserves the global database consistency. $\square$

Note that, in practice, many compensatable subtransactions do preserve global consistency individually. For instance, in Example 1, both t_1 and t_2 are compensatable. Moreover, they are globally consistent subtransactions.

Several issues relate to enforcing compensating serializability. Similarly to sagas [GMS87], from the point of an application programmer, a mechanism is required for informing the system of the beginning and end of a compensatable subtransaction of a flexible transaction that can independently reveal its results to other flexible transactions. As compensating serializability implies global serializability, at least the global serializability must be ensured. Much research of both a theoretical and a practical nature has been directed to maintaining global serializability [GRS91, BGMS92, MRB$^+$92, ZE93]. Many of the proposed approaches are applicable to our scenario. Moreover, a variation of the strict two-phase locking protocol can be designed at the global level to enforce the condition $WC(G_i^\natural) \cap RC(G_j) = \emptyset$ proposed in Definition 8. The main idea is to associate each data item with a global read lock and a global write lock at the global level. When a subtransaction of a flexible transaction wishes to access a data item at a local site, it must obtain a global lock on the data item from the GTM before this operation is submitted to the local site for execution. For the execution of those compensatable subtransactions which do not guarantee global consistency, their global write locks either must be held until all related subtransactions in the same partial order are committed, or they must be transferred to their compensating subtransactions. The discussion of the implementation details lies beyond this paper and is not presented here.

5 Conclusions

Global transaction management in an error-prone MDBS environment has been recognized as a substantial and as yet unresolved issue if the component local database systems do not support prepare-to-commit states. We have advanced a framework for flexible transaction management in the MDBS environment in which local database systems are required to maintain only serializability and recoverability. This framework includes the definition of the fundamental model and of the properties of flexible transactions, the classification of the flexible transactions that can be executed in the presence of failures, and the proposal of a new correctness criterion.

The most important properties of flexible transactions, namely, semi-atomicity and flex-isolation, have been precisely defined. By ensuring these properties, flexible transactions become more resilient to failures than the traditional global transactions. Also, more concurrency on the execution of flexible transactions can be achieved by releasing the partial effects of compensatable subtransactions prior to the commitment of the flexible transaction. Flexible transaction management is achieved by using compensation and retry approaches to ensure semi-atomicity and by maintaining compensating serializability on the concurrent execution of flexible transactions to ensure flex-isolation. Local prepare-to-commit states are thus not required. The construction of recoverable flexible transactions that are executable in the error-prone MDBS environment demonstrates that the flexible transaction model indeed enhances the scope of global transaction management beyond that offered by the traditional global transaction model.

References

[ANRS92] M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth. Using Flexible Transactions to Support Multi-System Telecommunication Applications. In *Proceedings of the 18th VLDB conference*, Vancouver, Canada, August 1992.

[ARNS92] M. Ansari, M. Rusinkiewicz, L. Ness, and A. Sheth. Executing Multidatabase Transactions. In *Proceedings of the 25th Hawaii International conference on System Sciences*, Hawaii, January 1992.

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

[BHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Databases Systems.* Addison-Wesley Publishing Co., 1987.

[BS88] Y. Breitbart and A. Silberschatz. Multidatabase update issues. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 135–142, June 1988.

[BST90] Y. Breitbart, A. Silberschatz, and G. Thompson. Reliable transaction management in a multidatabase system. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 215–224, May 1990.

[BSW88] C. Beeri, H. Schek, and G. Weikum. Multi-level transaction management, theoretical art or practical need? In *Proceedings of the International Conference on Extending Database Technology*, pages 134–154, March 1988.

[DE89] W. Du and A. Elmagarmid. Quasi Serializability: a Correctness Criterion for Global Concurrency Control in InterBase. In *Proceedings of the 15th International Conference on Very Large Data Bases*, pages 347–355, Amsterdam, The Netherlands, August 1989.

[DEK90] W. Du, A. Elmagarmid, and W. Kim. Effects of Local Autonomy on Heterogeneous Distributed Database Systems. Technical Report ACT-OODS-EI-059-90, MCC, February 1990.

[Eea92] Ahmed K. Elmagarmid and et. al. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann Publishers, 1992.

[ELLR90] A. Elmagarmid, Y. Leu, W. Litwin, and M. Rusinkiewicz. A Multidatabase Transaction Model for InterBase. In *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 507–581, Brisbane, Australia, August 1990.

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

[GMK88] H. Garcia-Molina and B. Kogan. Node Autonomy in Distributed Systems. In *Proceedings of the First International Symposium on Databases for Parallel and Distributed Systems*, pages 158–166, 1988.

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

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

[Gra81] J. Gray. The transaction concept: Virtues and limitations. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–154, Cannes, France, September 1981.

[GRS91] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. On serializability of multidatabase transactions through forced local conflicts. In *Proceedings of the 7th Intl. Conf. on Data Engineering*, pages 314–323, Kobe, Japan, April 1991.

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

[HR83] T. Haerder and A. Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4), July 1983.

[KLS90] H. Korth, E. Levy, and A. Silberschatz. A Formal Approach to Recovery by Compensating Transactions. In *Proceedings of the 16th International Conference on Very Large Data Bases*, Brisbane, Australia, August 1990.

[KPE92] Eva Kühn, Franz Puntigam, and Ahmed Elmagarmid. An execution model for distributed database transactions and its implementation in VPL. In A. Pirotte, C. Delobel, and G. Gottlob, editors, *Lecture Notes in Computer Science, Advances in Database Technology — EDBT '92*, pages 483–498. Springer-Verlag, 1992. Proceedings of the 3rd International Conference on Extending Database Technology, Vienna, Austria, March, 1992.

[LEB92] Y. Leu, A. Elmagarmid, and N. Boudriga. Specification and execution of transactions for advanced database applications. *Information Systems*, 17(2), 1992.

[Leu91] Y. Leu. *Flexible Transaction Management in the InterBase Project.* PhD thesis, Department of Computer Science, Purdue University, May 1991.

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

[LKS91a] E. Levy, H. Korth, and A. Silberschatz. An optimistic commit protocol for distributed transaction management. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Denver, Colorado, May 1991.

[LKS91b] 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*, August 1991.

[Mos81] J. E. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1981.

[MRB$^+$92] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. The concurrency control problem in multidatabases: Characteristics and solutions. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 288–297, 1992.

[MRKS92a] 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.

[MRKS92b] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. A transaction model for multidatabase systems. Technical Report TR-92-14, University of Texas at Austin Department of Computer Science, 1992.

[ÖV91] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, Inc., 1991.

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

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

[RELL90] M. Rusinkiewicz, A. Elmagarmid, Y. Leu, and W. Litwin. Extending the Transaction Model to Capture more Meaning. *ACM SIGMOD Record*, 19(1):3–7, March 1990.

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

[WV90] A. Wolski and J. Veijalainen. 2PC Agent method: Achieving serializability in presence of failures in a heterogeneous multidatabase. In *Proceedings of PARBASE-90*, Miami Beach, Florida, 1990.

[ZE93] Aidong Zhang and Ahmed K. Elmagarmid. A theory of global concurrency control in multidatabase systems. *The VLDB Journal*, July 1993.