Non-Serializable Executions in Heterogeneous Distributed Database Systems

Sharad Mehrotra
Rajeev Rastogi
Henry F. Korth
Abraham Silberschatz

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712-1188 USA

Abstract

The concept of serializability has been the traditionally accepted notion of correctness in database systems. However, in a heterogeneous distributed database system (HDBMS) environment, ensuring serializability is a difficult task mainly due to the desire of preserving the local autonomy of the participating local database systems. In this paper, we introduce a new correctness criterion for HDBMSs, two-level serializability (2LSR), and show that it preserves database consistency in several HDBMS models. Further, we present a simple protocol for ensuring that schedules in an HDBMS environment are 2LSR. This protocol is easily implementable and does not violate the local autonomy of sites.

1 Introduction

The problem of transaction management in a heterogeneous distributed database management system (HDBMS) has received considerable attention from the database community in recent years. The basic problem is to integrate a number of pre-existing local database management systems (DBMSs) located at different sites in a manner that allows transactions to access data residing at multiple sites, and at the same time preserve the local autonomy of the various sites.

Database consistency is traditionally ensured by requiring that the concurrent execution of transactions be serializable [10]. The problem of ensuring global serializability in an HDBMS has been studied extensively in recent years. However, proposed solutions result in a low degree of concurrency (e.g., [3], [6], [11]). The inherent difficulty in ensuring serializability in an HDBMS results from the fact that due to the local autonomy of sites, the global transaction manager (GTM) is unaware of indirect conflicts between global transactions caused due to the execution of local transactions. In such a scenario, in order to ensure serializability, the GTM has to take pessimistic measures (e.g., forcing direct conflicts among all global subtransactions as in [11]), thus resulting in a low degree of concurrency.

One way of increasing concurrency is to adopt a weaker correctness criterion than serializability for HDBMSs [2]. The requirements of a correctness criterion are that, 1) it should be easily implementable, and 2) schedules satisfying the criterion must preserve database consistency. Whether or not a correctness criterion preserves database consistency depends on the nature of the transactions and integrity constraints in the system. For example, the quasi-serializability (QSR) correctness criterion introduced in [5] preserves database consistency if the HDBMS model is restricted to disallow any inter-site integrity constraints besides replication, and global transactions do not have value dependencies (that is, execution of a global transaction at a site is independent of its execution at other sites).

In this paper, we identify various HDBMS models in which non-serializable executions preserve database consistency. We propose a new correctness criterion,
two-level serializability (2LSR), which is more general than quasi-serializability, and show that it preserves database consistency in the HDBMS models considered. A simple protocol for ensuring two-level serializability in HDBMSs that does not violate the local autonomy of sites is also presented. It is shown that the restrictions imposed on global transactions in [5] (absence of value dependencies between global subtransactions) in order to preserve database consistency can be relaxed if the GTM follows a variant of the protocol.

The paper contains a number of theorems and lemmas, whose proofs are of considerable length. For brevity, we have omitted these proofs. A formal development of the HDBMS model along with the proofs of the lemmas and theorems can be found in [9].

The remainder of the paper is organized as follows. In Section 2, we develop a model for HDBMS applications based on the partitioning of data items at each site. In Section 3, we define 2LSR, and develop HDBMS models in which 2LSR preserves database consistency. In Section 4, we present a protocol which ensures that schedules are 2LSR. In Section 5, we compare our work with related work. Concluding remarks are offered in Section 6.

2 The HDBMS Model

A heterogeneous distributed database consists of a set of autonomous pre-existing local databases $LDB_1$, $LDB_2$, $LDB_3$, ..., $LDB_m$ located at sites $s_1$, $s_2$, ..., $s_m$ respectively. Each $LDB_i$ consists of a set of data items $D_i$. We denote the set of all the data items in the heterogeneous distributed database by $D$, thus $D = \bigcup_{i=1}^{m} D_i$. We assume that the local databases are disjoint; that is, $D_i \cap D_j = \emptyset$, $i \neq j$.

A transaction is a sequence of read and write operations resulting from the execution of a transaction program. In an HDBMS there are two types of transactions:

- **Local transactions**, that execute at a single site. All pre-existing applications running at a local site before integration are local transactions.

- **Global transactions**, that may execute at more than one site. Global transactions are normally applications developed after the integration has been performed.

Each local database system has a local transaction manager (LTM) which is responsible for ensuring local database consistency. The global transaction manager (GTM), built on top of the existing databases, is responsible for ensuring global database consistency. The GTM controls the execution of global transactions. We assume that the interface between the GTM and the LTMs provides for operations to be submitted by the GTM to the LTMs, and the LTMs to acknowledge the completion of operations to the GTM. Also, each LTM is assumed to follow a concurrency control protocol that ensures serializability at local sites.

2.1 Local Autonomy

An important consideration in the design of the GTM software is the requirement that each local DBMS retains its local autonomy. In this paper, local autonomy is defined to mean:

- local sites are free to follow any concurrency control protocol to ensure local database consistency,

- LTMs do not communicate any information (e.g., conflict graph) relevant for concurrency control to the GTM, and

- local transactions execute outside the control of the GTM.

Local autonomy reflects the fact that the local DBMSs are independent and autonomous. This distinguishes the problem of transaction management in heterogeneous systems from that in homogeneous systems.

2.2 Integrity Constraints

Integrity constraints, denoted by $IC$, in a database, distinguish inconsistent database states from consistent ones. A database state $DS$ maps every data item in the database to a value in its domain. Thus, $DS$ can be expressed as a set of ordered pairs of data items and their values. A database state is consistent if it does not violate the integrity constraints. For example, consider a database consisting of data items $a$ and $b$, and $IC = \{a = b\}$. A database state $\{(a, 5), (b, 5)\}$ is a consistent database state. However, $\{(a, 5), (b, 6)\}$ is an inconsistent database state.

In an HDBMS, each local DBMS defines certain integrity constraints among the data items located at the site. As the local DBMSs are pre-existing and independent, there are no inter-site integrity constraints prior to integration. However, the integration of the various DBMSs into an HDBMS results in the introduction of certain global inter-site constraints (e.g., replicated data items). Various assumptions about
the kind of constraints introduced due to integration of the DBMSs can be made. For example, in [5], the
authors assume that the only kind of global integrity constraint that can be present is replication of data
over sites.

The observation that inter-site integrity constraints are introduced as a result of integration enables us to
partition the set of data items at a site, $D_i$, into local data items, $LD_i$, and global data items, $GD_i$, such
that $LD_i \cap GD_i = \emptyset$, and $D_i = LD_i \cup GD_i$. Furthermore, if there is an integrity constraint between $d_i \in LD_i$ and $d_j \in LD_j$, $i \neq j$, then $d_i \in GD_i$ and $d_j \in GD_j$. The set of all the global data items, $GD = \bigcup_{i=1}^{m} GD_i$, and the set of all the local data items, $LD = \bigcup_{i=1}^{m} LD_i$. Data items at different sites between which integrity constraints are introduced as a result of the integration are thus in $GD$. For example, all replicated data items are global (we model replicated data items as distinct data items at different sites with an equality constraint between them).

It must be noted that the definition of local and global data items suggests that there be no integrity
constraints between local and global data items at a site (as that could lead to an inter-site integrity con-straint involving local data items). While it is generally true that the presence of integrity constraints bet-ween local and global data items at a site may lead to an inter-site integrity constraint involving local data items, it is not necessary. For example, consider a database consisting of two site $s_1$ and $s_2$. Let $a \in LD_1$, $b \in GD_1$, and $c \in GD_2$. Let $IC = a > b \land c > b$. Note that there is an integrity constraint between local and global data items, but there is no inter-site integrity constraint involving the local data item $a$.

In an HDBMS, pre-existing local applications can be assumed to preserve the local database integrity constraints that were present prior to integration. However, they are unaware of the newly introduced integrity constraints. As a result, if local transactions were to write global data items, database consistency may not be preserved. For example, the resulting database state may be inconsistent if local transactions wrote replicated data items. Local transactions, thus, do not write global data items.

2.3 Global Transaction Programs

Since local applications are pre-existing, no assumptions can be made about the nature of the local transactions in the system. However, different assumptions about the nature of global transaction programs have been made by researchers. In [6], the authors assume that no value dependencies exist between subtransactions of a global transaction program, while in [3], no such assumption was made. A global transaction program has value dependencies if the operations performed by it at one site depend on the operations performed by it at other sites.

Example 1: Consider the following two transaction programs.

$$
t_{p_1} : \begin{cases}
\text{if}(act_1 \geq amt) & \text{then begin} \\
act_1 := act_1 - amt; & \text{act}_2 := act_2 + amt & \text{end}
\end{cases}
$$

$$
t_{p_2} : \text{return}(act_1 + act_2)
$$

Transaction program $t_{p_1}$ transfers money ($amt$) from account $act_1$ at site $s_1$ to account $act_2$ at site $s_2$, and has value dependencies. However, $t_{p_2}$, which returns the sum of the balances in $act_1$ and $act_2$, has no value dependency.

Requiring global transaction programs to have no value dependencies may be too restrictive for certain applications. Thus, we consider global transaction programs that are restricted to have fixed-structure. A global transaction program has fixed-structure if its execution from any database state results in the same transaction (however, the values read and written may depend on the database state). In Example 1, $t_{p_2}$ has fixed-structure, while $t_{p_1}$ does not. However, $t_{p_1}'$ which is a modification of $t_{p_1}$ and is shown below, has fixed-structure.

$$
t_{p_1}' : \begin{cases}
\text{if}(act_1 \geq amt) & \text{then begin} \\
act_1 := act_1 - amt; & \text{act}_2 := act_2 + amt & \text{end}
\end{cases}
$$

3 Two-Level Serializability (2LSR)

In this section, we introduce a new correctness criterion, two-level serializability (2LSR), for HDBMS envi-
ronments. A schedule $S$ is 2LSR if

- its projection on the set of global transactions is serializable (that is, deleting operations belonging to local transactions from $S$ results in a serializable schedule), and
• its projection on the set of data items at each site is serializable (that is, the schedules at each of the local sites is serializable).

The set of serializable schedules is a proper subset of the set of 2LSR schedules. An example of a 2LSR schedule that is not serializable is given in Example 2 below. In Example 2, as well as in all other examples in this paper, we use \( t_i \) to denote the transaction resulting from the execution of the transaction program \( t_{P_i} \). Operations belonging to transaction \( t_i \) are subscripted by \( i \), and include the value of the data item involved. Thus, a read operation on data item \( a \) belonging to transaction \( t_1 \) is denoted by \( r_{1}(a,v) \), where \( v \) is the value returned by the read.

Example 2: Consider an HDBMS consisting of two sites \( s_1 \) and \( s_2 \). Let \( D_1 = \{a,b\} \), \( D_2 = \{c,d\} \). Consider the following two global transaction programs.

\[
\begin{align*}
 t_{P_1} : & \quad a := c \\
 t_{P_2} : & \quad d := b
\end{align*}
\]

Let \( t_{P_3} \) and \( t_{P_4} \) be local transaction programs executing at sites \( s_1 \) and \( s_2 \), respectively.

\[
\begin{align*}
 t_{P_3} : & \quad b := a \\
 t_{P_4} : & \quad c := d
\end{align*}
\]

Consider the local schedules at sites \( s_1 \) and \( s_2 \) resulting from the execution of \( t_{P_1} \), \( t_{P_2} \), \( t_{P_3} \) and \( t_{P_4} \) from database state \( DS = \{(a,-1),(b,-1),(c,1),(d,1)\} \).

\[
\begin{align*}
 S_1 : & \quad r_{3}(a,-1) \quad \text{w}_{3}(a,1) \quad r_{4}(b,-1) \quad \text{w}_{4}(b,-1) \\
 S_2 : & \quad r_{1}(c,1) \quad \text{w}_{4}(c,1) \quad \text{w}_{3}(d,-1)
\end{align*}
\]

The above schedule is 2LSR, but not serializable. \( \Box \)

As noted earlier, schedules satisfying a correctness criterion must preserve database consistency; that is, they must not violate the integrity constraints of the database. However, this requirement does not prevent transactions in a schedule from "seeing" inconsistent database states. For example, schedules that are final-state serializable [10] may contain read only transactions that return inconsistent data item values. To overcome this, we define the notion of strong correctness. A schedule is said to be strongly correct if in addition to preserving the database integrity constraints, the values returned by the read operations of every transaction in the schedule are consistent.

We now develop a hierarchy of HDBMS models for which 2LSR schedules are strongly correct. These models are based on restricting the transactions’ read and write operations of the various data items. Our purpose in examining a variety of models is to explore the tradeoffs between various possible restrictions on transaction behavior. This provides insight into the competing goals of an HDBMS: local autonomy, data consistency, and generality of the allowed transactions.

We begin with a highly restrictive model (which we call the trivial model) and then proceed to less restrictive and more practical models.

3.1 The Trivial Model

Consider a model of HDBMS applications with the following restrictions on the transactions:

• local transactions only read and write local data items, and

• global transactions only read and write global data items.

For this model we can establish the following theorem.

Theorem 1: If \( S \) is a 2LSR schedule in the trivial model, then \( S \) is serializable and thus, is strongly correct. \( \Box \)

3.2 The Global Read (\( G_r \)) Model

Consider a model of HDBMS applications with the following restrictions on transactions:

• local transactions only read and write local data items, and

• global transactions in addition to reading and writing global data items, also read local data items.

The \( G_r \) model is suitable for HDBMS applications where global transactions are read-only queries. In the \( G_r \) model, 2LSR schedules may not preserve database consistency as is illustrated in Example 3.

Example 3: Consider an HDBMS consisting of two sites \( s_1 \) and \( s_2 \). Let \( D_1 = \{a,b,c\} \), \( D_2 = \{c\} \), \( LD = \{a\} \) and \( GD = \{b,c\} \). Let \( IC = \{a > 0 \rightarrow b > 0 \} \land \{c > 0 \rightarrow (b > 0 \lor e > 0)\} \). Note that \( (a > 0 \rightarrow b > 0) \) is an integrity constraint between local data item \( a \) and global data item \( b \). Consider the following global transaction programs.

\[
\begin{align*}
 t_{P_1} : \quad & \text{if}(a \leq 0) \quad \text{then} \quad e := 1 \\
 & \text{else} \quad e := -1; \\
 & e := 1
\end{align*}
\]
\[ tp_2: \begin{cases} \text{if}(a \leq 0) & b := -1 \\ \text{else} & b := 1 \end{cases}; \]
\[ c := -1 \]

Let \( tp_3 \) be a local transaction program executing at site \( s_1 \).

\[ tp_3: \begin{cases} a := -1 \end{cases} \]

Consider the local schedules at sites \( s_1 \) and \( s_2 \) resulting from the execution of \( tp_1, tp_2 \) and \( tp_3 \) from the database state \( DS = \{(a, 1), (c, 1), (b, 1), (e, 1)\} \).

\( S_1: r_1(a, 1) w_1(c, -1) w_2(a, -1) r_2(a, -1) w_2(b, -1) \)
\( S_2: w_2(c, -1) w_1(c, 1) \)

The final global database state resulting from the above schedule is \( \{(a, -1), (c, 1), (b, -1), (e, -1)\} \), which is inconsistent. \( \square \)

In Example 3 above, note that there is an integrity constraint between \( a \) which is a local data item and \( b \) which is a global data item. If we further restrict the model and disallow such integrity constraints between local and global data items, 2LSR schedules can be proved to be strongly correct.

**Theorem 2:** Let \( S \) be a 2LSR schedule in the \( G_r \) model. If no integrity constraints are present between local and global data items, then \( S \) is strongly correct. \( \square \)

### 3.3 The Local Read (\( L_r \)) Model

Consider a model of HDBMS applications with the following restrictions on transactions:
- local transactions read and write local data items and also read global data items, and
- global transactions only read and write global data items.

As in the \( G_r \) model, 2LSR schedules may not always preserve database consistency in the \( L_r \) model as the following example illustrates.

**Example 4:** Consider an HDBMS consisting of two sites \( s_1 \) and \( s_2 \). Let \( D_1 = \{b, c, e\} \), \( D_2 = \{a\} \), \( LD = \{e\} \) and \( GD = \{a, b, c\} \). Let \( IC = (a \cdot c \cdot e > 0) \wedge (b > 0 \rightarrow c > 0) \wedge e > 0 \). Thus, no integrity constraints are present between local and global data items. Consider the following two global transaction programs.

\[ tp_1: \begin{cases} b := 1; \\ \text{if}(a \leq 0) & c := 1 \\ \text{else} & c := 1 \end{cases} \]
\[ a := 1 \]

Let \( tp_3 \) be a local transaction program executing at site \( s_1 \).

\[ tp_3: \begin{cases} \text{if}(b > 0) & e := c \\ \text{else} & e := 1 \end{cases} \]

Consider the local schedules at sites \( s_1 \) and \( s_2 \) resulting from the execution of \( tp_1, tp_2 \) and \( tp_3 \) from the database state \( DS = \{(a, -1), (b, -1), (c, -1), (e, 1)\} \).

\( S_1: w_1(b, 1) r_2(b, 1) r_3(b, -1) w_2(c, 1) w_3(e, -1) \)
\( S_2: w_2(a, 1) r_1(a, 1) \)

The final state resulting from the execution of the above schedule is \( \{(a, 1), (b, 1), (c, 1), (e, -1)\} \), which is inconsistent. \( \square \)

In Example 4, the local transaction program \( tp_3 \) reads inconsistent data values. The reason for the reads being inconsistent is that \( tp_1 \) leaves the database at site \( s_1 \) in an inconsistent state, which is then "seen" by \( tp_3 \). However, if we restrict the transaction programs to have no value dependencies, then 2LSR schedules can be shown to be strongly correct.

**Theorem 3:** Let \( S \) be a 2LSR schedule in the \( L_r \) model. If global transaction programs do not have value dependencies, then \( S \) is strongly correct. \( \square \)

It must be noted that Theorem 3 holds even if integrity constraints are present between local and global data items.

### 3.4 The Global Read-Write (\( G_{rw} \)) Model

Consider a model of HDBMS applications with the following restrictions on transactions:
- local transactions only read and write local data items, and
- global transactions, in addition to reading and writing global data items, also read and write local data items.

The \( G_{rw} \) model is more general than the \( G_r \) model. Thus, as shown in Example 3, in the \( G_{rw} \) model, 2LSR schedules may not preserve database consistency if integrity constraints are present between local and
global data items. However, in contrast to the $G_r$ model, absence of integrity constraints between local and global data items does not ensure database consistency as is demonstrated by the following example.

**Example 5:** Consider an HDBMS consisting of two sites $s_1$ and $s_2$. Let $D_1 = \{a, b, c, e\}$, $D_2 = \{c\}$, $LD = \{a, b, c, e\}$, and $GD = \{\}$. Let $IC = (a > 0 \rightarrow b > 0) \land c > 0 \land e > 0$. Thus, no integrity constraints are present between local and global data items. Consider the following two global transaction programs.

$$tp_1: \begin{cases} \text{if}(a > 0) & \text{then} \ c := b \\ \text{else} & \ c := 1 \end{cases}$$

$$tp_2: \ e := c$$

Let $tp_3$ be a local transaction program executing at site $s_1$.

$$tp_3: \ a := 1;$$

$$\begin{cases} \text{if}(c > 0) & \text{then} \ b := 1 \end{cases}$$

Consider the local schedules at sites $s_1$ and $s_2$ resulting from the execution of $tp_1, tp_2$ and $tp_3$ from the database state $DS = \{(a, \bar{1}), (b, \bar{1}), (c, \bar{1}), (e, \bar{1})\}$. 

$$S_1 : w_1(a, 1) \ r_1(a, 1) \ r_1(b, -1) \ w_2(e, -1) \ r_3(e, -1)$$

$$S_2 : w_2(c, -1) \ r_2(c, -1)$$

The final state resulting from the execution of the above schedule is $\{(a, \bar{1}), (b, -1), (c, -1), (e, -1)\}$, which is inconsistent. □

In Example 5, database inconsistency results from the fact that transaction program $L$ does not have fixed-structure.

**Theorem 4:** Let $S$ be a 2LSR schedule in the $G_{rw}$ model resulting from the execution of transaction programs with fixed-structure. If no integrity constraints are present between local and global data items, then $S$ is strongly correct. □

Note that even local transaction programs are required to have fixed-structure, if database consistency is to be preserved. In Example 5, only the local transaction program did not have fixed-structure and that resulted in an inconsistent database state. Since local transaction programs are pre-existing, they may not meet the requirement of having fixed-structure. Thus, applicability of Theorem 4 is limited. Another way of ensuring that 2LSR schedules are strongly correct is to restrict global transaction programs to have no value dependencies. In the next subsection, it is shown that, under this restriction, 2LSR schedules are strongly correct for the $G_{rw \ L_r}$ model, which is more general than the $G_{rw}$ model.

### 3.5 The Global Read-Write and Local Read ($G_{rw \ L_r}$) Model

Consider a model of HDBMS application with the following restriction on transactions.

- local transactions read and write local data items and also read global data items, and
- global transactions read and write global and local data items.

The $G_{rw \ L_r}$ model is more general than all the models considered so far. Since the $G_{rw \ L_r}$ model is more general than the $G_r$ model, presence of integrity constraints between local and global data items may result in the violation of database consistency as was illustrated in Example 3. Similarly, as shown in Example 4, database consistency may be violated if global transaction programs have value dependencies. The following theorem states conditions under which 2LSR schedules preserve database consistency in the $G_{rw \ L_r}$ model.

**Theorem 5:** Let $S$ be a 2LSR schedule in the $G_{rw \ L_r}$ model. If global transaction programs do not have value dependencies, and no integrity constraints are present between local and global data items, then $S$ is strongly correct. □

### 4 Concurrency Control Protocols

In order to ensure that a schedule $S$ is two-level serializable, the GTM protocol only needs to guarantee that the projection of $S$ on the set of global transactions is serializable. This follows from the assumption that the LTM at each site ensures the serializability of the projection of $S$ on the data items at the site. Thus, in an HDBMS, two-level serializability can easily be ensured since the global transactions execute under the control of the GTM. For example, if the GTM maintains global locks, then the following GTM protocol ensures that schedules are 2LSR.

- Global transactions follow the 2PL protocol while obtaining and releasing global locks.
- A global lock is held by a global transaction at least until the completion of the operation, at the local site, for which the lock was obtained.
Let us consider a variant of the above scheme. Suppose that each LTM follows the 2PL protocol, and the global transactions, in addition to obtaining and releasing global locks in a 2PL fashion, do not release a global lock until all the locks requested by the global transaction have been obtained at the local sites. We refer to the schedules obtained as a result of following the above scheme as two-level two-phase locking (2LPL) schedules.

Every 2LPL schedule is a 2LSR schedule but not vice-versa. This is illustrated by the schedule in Example 5 which is 2LSR but not 2LPL. In the $L_r$ and $G_r L_r$ models, for 2LSR schedules to preserve database consistency, global transaction programs are required to have no value dependencies. However, if schedules are 2LPL, this restriction can be relaxed, and database consistency is preserved in these models even if transaction programs have value dependencies.

**Theorem 6:** Let $S$ be a 2LPL schedule in the $G_r L_r$ model. If there are no integrity constraints between local and global data items, and global transaction programs have fixed-structure, then $S$ is strongly correct. □

As the $G_r L_r$ model is a generalization of the $L_r$ model, the following corollary trivially follows.

**Corollary 1:** Let $S$ be a 2LPL schedule in the $L_r$ model. If there are no integrity constraints between local and global data items, and global transaction programs have fixed-structure, then $S$ is strongly correct. □

The importance of Theorem 6 is the implication that no restrictions need to be imposed on local transaction programs as we require only global transaction programs to have fixed-structure. Thus, the local autonomy of sites is not violated. However, it is necessary for the global transaction programs to have fixed-structure, since otherwise the hypothesis of Corollary 1, and hence Theorem 6, do not hold. This is illustrated by Example 4, where the schedule is a 2LPL schedule, but since transaction program $t_1$ does not have fixed-structure, database consistency is not preserved.

---

5 If the LTM interface does not provide for the submission of explicit lock operations, then the GTM can utilize the acknowledgements for operations to determine if local locks have been obtained.

5 Related Work

Most of the work on transaction management in HDBMSs has been concentrated on ensuring global serializability, and has not addressed the issue of non-serializable executions that preserve database consistency. An exception is the work by Du and Elmagarmid in which the notion of quasi serializability (QSR) was introduced [5]. In this section, we discuss the relation of our work to QSR.

**Definition 1:** A schedule $S$ is quasi serial if and only if local schedules are serializable and there is a total order of global transactions such that for any two global transactions $t_i$ and $t_j$, if $t_i$ precedes $t_j$ in the total order, then all of $t_i$'s operations precede all of $t_j$'s operations at each and every local site. A schedule $S$ is QSR if it is conflict equivalent to a quasi serial schedule. □

Since in every QSR schedule $S$, the projection of $S$ on the set of global transactions is serializable, QSR schedules are a subset of 2LSR schedules. Further, the schedule in Example 3 is a 2LSR schedule which is not QSR. Thus, QSR schedules are a proper subset of 2LSR schedules.

In [5], the authors claim that adopting QSR as a correctness criterion in an HDBMS environment requires that:

1. there be no integrity constraints between data items at different sites (except equality constraints which model replicated data items),
2. transaction programs have no value dependencies, and
3. local transactions not be permitted to write replicated data items.

QSR schedules can be shown to preserve database consistency within the framework that we have developed for studying transaction management in HDBMSs. To illustrate this, suppose that we further restrict the $G_r L_r$ model as follows:

1. Global data items consist only of replicated data items across sites.
2. No integrity constraints are present between local and global data items (presence of such integrity constraints would result in inter-site integrity constraints).
3. Transaction programs do not have value dependencies.
Using Theorem 5, the result that QSR schedules preserve database consistency under restrictions mentioned in [5] directly follows.

Theorem 5, however, states a more general result, in that restriction 1 mentioned above is not required, and schedules are required to be 2LSR, not QSR. Further, if schedules are 2LPL, then the requirement that no value dependencies be present global subtransactions can be relaxed as shown in Theorem 6.

6 Conclusion

In this paper, we adopt a weaker correctness criterion than serializability for HDBMS applications, which we refer to as two-level serializability (2LSR). The motivation for abandoning serializability as the correctness criterion is the low degree of concurrency that results from protocols for ensuring serializability in HDBMSs. Two level serializability only requires the projection of the global schedule on the set of global transactions to be serializable and each of the local schedules to be serializable. As a result, protocols for ensuring that schedules are 2LSR have the advantages of being simple, allowing a high degree of concurrency, and not violating the local autonomy of sites.

However, 2LSR schedules preserve database consistency only in certain restricted HDBMS applications. In many HDBMS applications, the data items for which intersite integrity constraints are introduced due to the integration of DBMSs are known (e.g., replicated data). We capture this knowledge in our HDBMS model by partitioning data items into two disjoint sets: global and local data items. A data item is a global data item if there is an integrity constraint between it and a data item at a different site. This knowledge allows us to prove, in certain cases, that non-serializable executions preserve database consistency. We identified models for several of these cases, each of which involves different restrictions on a transaction's read and write operations. The models provide a range of options to the designer of an HDBMS. We have characterized the relative power of our models, both in terms of concurrency and restrictions imposed on transactions.

Our approach is only the first step in relaxing the serializability requirement in an HDBMS environment. In certain HDBMS applications, knowledge of the exact nature of the inter-site integrity constraints may be available. It may be possible to exploit the knowledge of the integrity constraints to further relax the restrictions on the global transactions and develop more HDBMS models for which 2LSR preserves database consistency. Further, fault tolerant algorithms for transaction management in HDBMS applications with these new correctness criteria still need to be developed.

References


