1989

Transaction Accessibility and Integrity Issues in Heterogeneous Distributed Database Systems

Weimin Du

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

Marek Rusinkiewicz

Report Number:
89-924

TRANSACTION ACCESSIBILITY AND INTEGRITY ISSUES IN HETEROGENEOUS DISTRIBUTED DATABASE SYSTEMS

Weimin Du
Ahmed K. Elmagarmid
Marek Rusinkiewicz

CSD-TR-924
October 1989

Transaction Accessibility and Integrity Issues in Heterogeneous Distributed Database Systems (Extended Abstract)[1]

Weimin Du, Ahmed K. Elmagarmid and Marek Rusinkiewicz
Computer Sciences Department
Purdue University
West Lafayette, IN 47907
(317)-494-1998
ahmed@cs.purdue.edu

[1]This work is supported by a PYI Award from NSF under grant IRI-8857952 and grants from AT&T Foundation, Tektronix, and Mobil Oil.

1 Introduction

A heterogeneous distributed database system (HDDBS) is a federation of pre-existing and autonomous databases. An HDDBS is different from a homogeneous distributed database system in that its local database systems (LDBSs) are autonomous.

Traditionally, a database system is modeled, from the view point of transaction management, as a set of data and a set of transactions accessing the data. This basic model, although very simple, is powerful enough to be used to develop serializability theory, the standard model for database concurrency control.

It has been argued that maintaining global serializability in HDDBSs is very difficult [DELO89]. In cases where global serializability can be maintained, the cost of doing so is very high. This motivated studies of new correctness criteria for global concurrency control in HDDBSs, e.g., quasi serializability [DE89a]. Such a criterion is suitable for HDDBSs because it takes into account the fact that LDBSs might be autonomous. The appropriateness of these new criteria are not well demonstratable using existing models. This is mainly because these models are unable to incorporate autonomy requirements present in HDDBs.

Access to data may be restricted due to various reasons, e.g., security authorization may be limited to a particular segment of a LDBS. This restriction can be modeled using transaction accessibility. Restated in terms of autonomy, each LDBS determines independently what parts of the data it owns are accessible to what class of transactions. A LDBS may limit access to its data by global transactions to read only for example. This notion defines a limited transaction accessibility. This type of transaction accessibility is referred to as static transaction accessibility (STA). Interestingly, STA has significant effects on HDDB integrity. This is due to the restrictions imposed by STA. Since STA limits arbitrary requests made by transactions on data, in turn, it simplifies the range of integrity constraints that have to be enforced.

STA represents the static semantics of an HDDBS, and therefore can be incorporated into the database system model. In this paper, we present such an extended model. In order to be able to model HDDBSs with various STAs, transactions are categorized based on functionality (i.e., local, distributed, or global) and data are categorized based on accessibility (i.e.,

what transactions are allowed to access what portions of the data). This model is then formalized and an appropriateness proof of quasi serializability is given.

The problem of transaction accessibility has been studied in [KGM87] to develop a new correctness criterion, namely virtual serializability. Our work is different from theirs in the following two ways.

1. We specify transaction accessibility at data level, making our transaction accessibility model more powerful.
2. We provide a framework for validating new criteria by incorporating STA into the database system model.

The primary contributions of the paper are twofolds.

1. Introduce a new HDDBS model augmented with STA.
2. Show how to use the new model to study the integrity problem.

The remainder of this paper is organized as follows. We first introduce, in section 2, the new database system model. Then we discuss, in section 3, the effects of STA on the HDDBS integrity problem using two examples. In section 4, we briefly discuss how the second type of transaction accessibility, namely dynamic transaction accessibility can be used to preserve integrity constraints. Conclusions and a few remarks are given in section 5.

2 An HDDBS Model with Static Transaction Accessibility

Formally, an HDDBS can be modeled as a triple $< D, T, A >$, where D is the set of data stored in the HDDBS, T is the set of transactions executed in the HDDBS, and A is the specification of static accessibilities of transactions in T to data in D.

Before showing how STAs are specified, we give a classification of transactions and data. The specification of STA is based on this classification.

Transactions

In our model, transactions are classified into one of the following three categories:

- **A local transaction** is initiated by a local database management system (LDBMS) and accesses data residing at that site.
- **A distribute transaction** is also initiated by a LDBMS and accesses data residing at that site and other sites.
- **A global transaction** is initiated by the global database management system (GDBMS) and accesses data residing at more than one site.

A local transaction may be either flat or nested. It is nested if its host LDBS is distributed. Global and distributed transactions, however, are always nested because they access more than one LDBS.

Let GT be the set of global transactions, DT_i the set of distribute transactions initiated at site i, and LT_i the set of local transactions executed at site i, then we have $T = GT \cup DT_1 \cup LT_1 \cup ... \cup DT_n \cup LT_n$. Transactions in different groups have different accessibilities.

Data

The entire body of data in an HDDBS is physically divided into $n + 1$ subsets (or sites), denoted $S_0, S_1, ..., S_n$. The subset S_i $(1 < i < n)$ residing at LDBS_i is managed by LDBMS_i. While the subset S_0 is managed by the GDBMS. The data can also be viewed logically as belonging to one of the following six categories. This logical classification is based on the availability of data to various transactions.

- **Type 0:** Data of this type is readable and updatable by global transactions only.
- **Type 1:** Data of this type is readable by all global and distributed transactions, as well as the local transactions executing at the site. However, they are updatable by global transactions only.
- **Type 2:** Data of this type is readable by all global and distributed transactions, as well as the local transactions executing at the site. However, they are updatable by global and distributed transactions only.

- **Type 3:** Data of this type is updatable by all global and distributed transactions, as well as the local transactions executing st the site.
- **Type 4:** Data of this type is readable by all global and distributed transactions, as well as the local transactions executing st the site. However, they are updatable by the local and distributed transactions initiated at the site only.
- **Type 5:** Data of this type is accessible to only those local and distributed transactions that are initiated at the site.

The above classification is depicted in Figure 1.

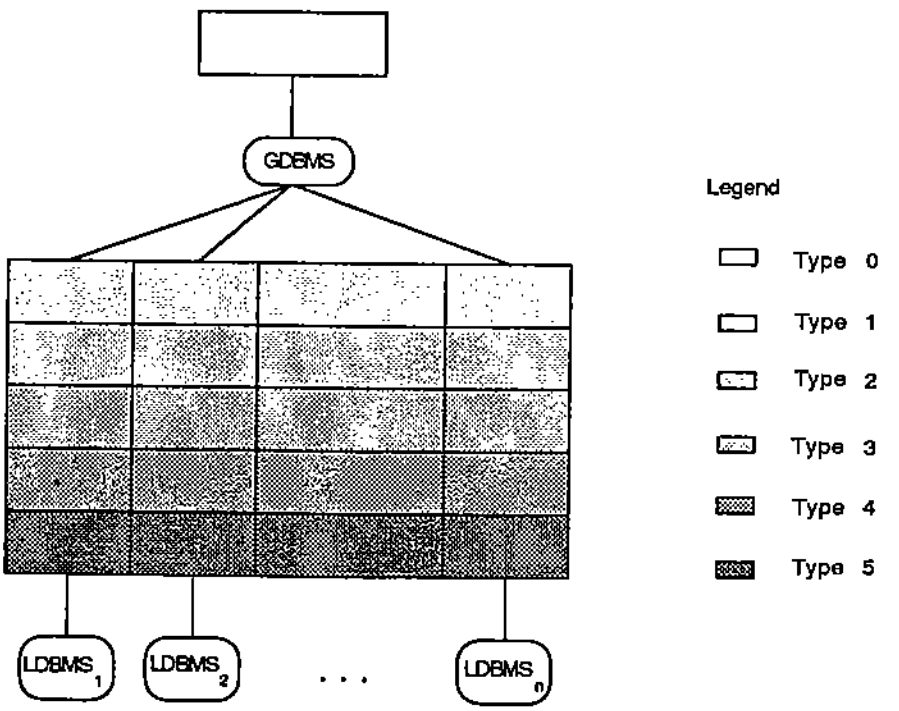

Figure 1: A classification of data in an HDDBS

The difference between type 2 and 3 data is that the former is usually replicated because it is frequently requested by other sites, while the latter is updatable by local transactions and, therefore, cannot be replicated[1].

[1]In our model, replicated data is updatable by global and distributed transactions

Let us consider, as an example, an HDDBS for university employees. The HDDBS consists of several LDBSs run by individual departments. The dean's office where the GDBMS is located is in charge of the consistency of the global database. The following are examples of various types of data.

- **Type 0:** Dean's comments
- **Type 1:** Statistical data
- **Type 2 and 3:** Home addresses, telephone number
- **Type 4:** Courses currently teaching, research projects
- **Type 5:** Salaries, department evaluations

The above classification represents general requirements on data in HDDBSs. A specific HDDBS may not have all six types of data. For example, if no local data (e.g., home addresses) is replicated, there will be no type 2 data.

The difference between global and distributed transactions is their ability to access type 0 and type 1 data. For example, an individual department is not eligible to maintain the statistical data because it may not have access to all related data. Distributed transactions issued by departments should therefore not allowed to update these statistical data. The distinction between global and distributed transactions is important because their executions effect different types of integrity constraints, as shown in the next section.

Static Transaction Accessibility

Let $H =< D, T, A >$ be an HDDBS. A is illustrated using matrix formalism as follows.

Definition 2.1 (Transaction accessibility matrix:) *Given $t \in T$, the transaction accessibility matrix (TAM) of t is a $6 \times n$ matrix $M(t)$ where n*

only. Once a copy of replicated data is updated, all its replicas should be updated accordingly. The reason for doing so is that update propagation is difficult, if not impossible, in HDDBSs because of autonomy.

is the number of LDBSs in H and the elements of M(t) are defined as follows.

$$M(t)[i,j] = \begin{cases} N & \textit{if } t \textit{ has no access to type } i \textit{ data at site } j \\ R & \textit{if } t \textit{ can read type } i \textit{ data at site } j \\ W & \textit{if } t \textit{ can both read and write type } i \textit{ data at site } j \end{cases}$$

We assume that the ability of updating a data implies that of reading the data. Let us use " $<$ " to represent this relation. We have,

$$N < R < W$$

Let $t_1, t_2 \in T$ and $\mathrm{M}(t_1)$ and $\mathrm{M}(t_2)$ be their TAMs, respectively. The inclusiveness relation of two TAMs is defined as follows.

1. $\mathrm{M}(t_1) < \mathrm{M}(t_2)$ if $\mathrm{M}(t_1)[i,j] < \mathrm{M}(t_2)[i,j]$ for all $i = 1,2,...,6$ and $j = 1,2,...,n$.
2. $\mathrm{M}(t_1) \leq \mathrm{M}(t_2)$ if either $\mathrm{M}(t_1)[i,j] < \mathrm{M}(t_2)[i,j]$ or $\mathrm{M}(t_1)[i,j] = \mathrm{M}(t_2)[i,j]$ for all $i = 1,2,...,6$ and $j = 1,2,...,n$.

Let $a_1, a_2 \in \{N, R, W\}$. The sum of a_1 and a_2, denoted $a = a_1 \oplus a_2$, is defined as follows.

$$a = \begin{cases} a_1 & \text{if } a_2 \leq a_1, \\ a_2 & \text{otherwise.} \end{cases}$$

The sum of two TAMs is represented as $T = M(t_1) \oplus M(t_2)$, where $\mathrm{T}[i,j] = \mathrm{M}(t_1)[i,j] \oplus \mathrm{M}(t_2)[i,j]$. The sum operation is associative and commutative.

Definition 2.2 (Static transaction accessibility of HDDBSs:) *The STA of H is a* $2 * n + 1$ *tuple* $A = < GA, DA_1, LA_1, ..., DA_n, LA_n >$, *where,* $GA = \oplus_{t \in GT} M(t)$, $DA_i = \oplus_{t \in DT_i} M(t)$, *and* $LA_i = \oplus_{t \in LT_i} M(t)$ *for* $i = 1,2,...,n$.

Let $A_0 = < GA', DA'_1, LA'_1, ..., DA'_n, LA'_n >$, where for $i = 1,2,...,n$

$$DA'_i = \begin{pmatrix} & & & \overbrace{\quad} \text{ith column} & & \\ N & N & ... & N & ... & N \\ R & R & ... & R & ... & R \\ W & W & ... & W & ... & W \\ W & W & ... & W & ... & W \\ R & R & ... & W & ... & R \\ N & N & ... & W & ... & N \end{pmatrix}$$

$$LA_i' = \begin{pmatrix} & & & \text{ith column} & & \\ & & & \overbrace{\quad} & & \\ N & N & ... & N & ... & N \\ N & N & ... & R & ... & N \\ N & N & ... & R & ... & N \\ N & N & ... & W & ... & N \\ N & N & ... & W & ... & N \\ N & N & ... & W & ... & N \end{pmatrix}.$$

and

$$GA' = \begin{pmatrix} W & W & ... & W \\ W & W & ... & W \\ W & W & ... & W \\ W & W & ... & W \\ R & R & ... & R \\ N & N & ... & N \end{pmatrix}$$

Then we have,

Theorem 2.1 *For any HDDBS* $< D, T, A >$ *where* $A =< GA, DA_1, LA_1, ..., DA_n, LA_n >$, $GA \leq GA'$ *and* $DA_i \leq DA_i'$ *and* $LA_i \leq LA_i'$ *for* $i = 1, 2, ..., n$.

The above specification of STA has the following advantages.

1. It is formal. In other words, the STA of an HDDBS can be quantitatively specified.
2. It is powerful. The STA of an HDDBS is modeled at a more detailed level than site.

These advantages make it a suitable tool for studying the integrity problem of HDDBSs.

3 Effects of STA on Database Integrity: Case Studies

Database integrity is one aspect of database consistency, and is specified by a set of integrity constraints. It is usually assumed that a transaction,

when executing alone, preserves these integrity constraints. Therefore, a serializable execution also preserves database integrity. In HDDBSs, it is also assumed that subtransactions of global/distributed transactions preserve the integrity constraints of LDBSs if they execute in isolation.

In HDDBSs, integrity constraints are classified into one of the following categories.

- **Local integrity constraints** are defined on type 2, 3, 4 and 5 data of a single LDBS. They are the pre-existing integrity constraints for that LDBS. A sufficient condition for preserving local integrity constraints at a site is the serializability of local executions at that site.

- **Global integrity constraints** are defined on type 0 and 1 data. These are new integrity constraints specified after the integration process. Since type 0 and 1 data is updatable by global transactions only, serial execution of global transactions is sufficient to preserve global integrity constraints.

- **Distributed integrity constraints** are defined on type 2 and 3 data of several LDBSs. They are generally required because the underlying data is related by global/distributed transactions. Since all three kinds of transactions can update the data, the only general way of preserving distributed integrity constraints is to execute all transactions in a globally serializable fashion.

The above constraints form all possible integrity constraints in an HDDBS. Other kinds of integrity constraints are neither reasonable nor preservable. For example, there is no integrity constraints on type 4 data at different sites. The reason is that there is no relationship between these data due to local (design) autonomy. Therefore, it does not make any sense to impose constraints on them. In addition, these constraints, even if they were specified, will not be preserved by a local transaction simply because it can not access related data at other sites.

Given an HDDBS, we say that its global database integrity is preserved if all the three kinds of integrity constraints are attainable.

We now discuss how the HDDBS integrity is preserved by imposing restrictions on STA of the HDDBS. We do so using two case studies: global/ distributed read only HDDBSs and disjointly updatable HDDBSs. They are

chosen because (1) they represent practical HDDBS scenarios, and (2) their integrity constraints can be effectively preserved (e.g., by local serializable or quasi serializable executions).

3.1 Case 1: No Global/Distributed Update

In this subsection, we study the database integrity problem for those HDDBSs that limit global and distributed transactions to read only.

Definition 3.1 (Global/distributed read only HDDBSs) *An HDDBS* $H =< D, T, A >$ *is global/distributed read only if* $A =< GA, DA_1, LA_1, ..., DA_n, LA_n >$, *and*

$$GA \leq \begin{pmatrix} R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ N & N & ... & N \end{pmatrix} \quad and \quad DA \leq \begin{pmatrix} N & N & ... & N \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \end{pmatrix}$$

where $DA = \oplus_{i=1}^{n} DA_i$

HDDBSs that allowed queries only have existed for years. However, it is not very clear if these global/distributed read only HDDBSs maintain global database consistency and what kind of consistency they preserve. For example, a global execution in a global read only HDDBS may be non-serializable [DELO89].

The following theorem shows that local serializability is a sufficient condition for global/distributed read only HDDBSs.

Theorem 3.1 *Let* H *be a global/distributed read only HDDBS. The global database integrity of* H *is preserved if all local executions are serializable.*

An informal proof of the theorem is sketched as follows.

- **Case 1:** Local integrity constraints are always preserved because of the serializability of local executions.

- **Case 2:** Global integrity constraints are also preserved because type 0 and 1 data are not updatable.
- **Case 3:** Distributed integrity constraints will never be violated by any execution of global and distributed transactions. A local transaction, when it executes alone, will preserve distributed integrity constrains. In addition, the execution of a local transaction at a site is totally independent of executions at other sites. Therefore, distributed integrity constraints will not be violated by any execution of local transactions as long as it is locally serializable.

Our restriction on global transactions can be relaxed. For example, we may allow global transactions to update type 0 data. This will not affect local and distributed integrity constraints. In order to preserve global integrity constraints, global transactions should, however, access type 0 data in a serializable way.

3.2 Case 2: No Data Updated by Both Global and Distributed Transactions

Another case we shall study is characterized by the restriction of disallowing data updated by both global and distributed/local transactions.

Definition 3.2 (Disjointly updatable HDDBSs) *An HDDBS* $H =< D, T, A >$ *is disjointly updatable if* $A =< GA, DA_1, LA_1, ..., DA_n, LA_n >$ *and*

$$GA \leq \begin{pmatrix} W & W & ... & W \\ W & W & ... & W \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ N & N & ... & N \end{pmatrix}$$

and

$$DA \leq \begin{pmatrix} N & N & ... & N \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ W & W & ... & W \\ W & W & ... & W \end{pmatrix} \quad LA \leq \begin{pmatrix} N & N & ... & N \\ R & R & ... & R \\ R & R & ... & R \\ R & R & ... & R \\ W & W & ... & W \\ W & W & ... & W \end{pmatrix}$$

where $DA = \oplus_{i=1}^{n} DA_i$ *and* $LA = \oplus_{i=1}^{n} LA_i$.

In a disjointly updatable HDDBS, distributed transactions are not allowed to update data at other sits. The data global transaction can update (type 0 and 1) and the data distributed/local transactions at each site can update (type 4 and 5) are disjoint.

The disjointly updatable HDDBS model is more powerful than the global/distributed read only model in the following two ways.

1. Global/distributed transactions can update some data.
2. Changes made by global/distributed transactions can be seen by other distributed/local transactions.

A nice property of a disjointly updatable HDDBS is that its global database integrity can be preserved effectively.

Theorem 3.2 *Let H be a disjointly updatable HDDBS. The global database integrity of H is preserved if the global execution is quasi serializable [DE89a].*

An execution is quasi serializable if (1) all its local executions are serializable, and (2) it is equivalent to a quasi serial execution in which global and distributed transactions are executed sequentially.

To see why a quasi serial execution (and therefore a quasi serializable execution) preserves the global database integrity of a disjointly updatable HDDBS, let us consider the following three cases.

- **Case 1:** Local integrity constraints are preserved because of the serializability of local executions.
- **Case 2:** Global integrity constraints are also preserved because global transactions are executed sequentially.
- **Case 3:** Since type 2 and 3 data are read only, integrity constraints defined on them, if any, will always be preserved.

4 Using Dynamic Transaction Accessibility

Dynamic accessibility of a transaction is the ability of the transaction to actually access a portion of data that is statically available to it. The dynamic accessibility of a transaction is determined at the time when the transaction is written and submitted. Therefore, it can not be used in general to simplify the database consistency problem. However, it can be used in specific to simplify global transaction management.

Due to local autonomy, the dynamic accessibility of local transactions is usually not available to the GDBMS. This makes it impossible for the GDBMS to make use of dynamic transaction accessibility at any level lower than site. At the site level, on the other hand, dynamic accessibility of global/distributed transactions can be used to predict possible conflicts at the local level.

A useful tool to analyze dynamic transaction accessibility is the dynamic access graph[2] (DAG). A DAG for a global/distributed transaction t is a graph DAG(t) = $< N, E >$, where $N \subseteq \{S_0, S_1, ..., S_n\}$ is the set of names of the sites accessed by t, and E is the set of edges that connect nodes in N to form an acyclic graph. Given a global execution, its DAG is the union of DAGs for all global and distributed transactions in the execution. Given an HDDBS and a global execution E over it, the global database consistency of the HDDBS is preserved if there exists an acyclic DAG for E.

This result can be used to improve our result in theorem 3.2 as follows.

Given a disjointly updatable HDDBS and a global execution E over the HDDBS. Suppose that DAG(E) is cyclic. Let $GT = GT' \cup GT''$ and $DT = DT' \cup DT''$, where $GT' \cup DT'$ consists of all global and distributed transactions that are not involved in any cycle in DAG(E).

The fact that the transactions in $GT' \cup DT'$ are not involved in any cycle in DAG(E) implies that their executions will always preserves the database integrity constraints. Therefore, they may be ignored in the global concurrency control as far as the database integrity is concerned.

Theorem 4.1 *The global database integrity of the HDDBS is preserved by E if E is equivalent to an execution E' in which*

[2]The notion of DAG was introduced and studied in [BS88] and was originally called site graph

1. *all local executions are serializable; and*

2. *transactions in $GT'' \cup DT''$ execute sequentially at all sites (including S_0).*

5 Conclusion

In this paper, we have extended the basic database system model to incorporate STA. This model is useful in studying the integrity problem of HDDBSs with various autonomy requirements. We have also shown, by examples, how to simplify the HDDBS integrity problem by imposing restrictions on STA. In particular, we have demonstrated the appropriateness of quasi serializability as a correctness criterion in disjointly updatable HDDBSs. We also discussed very briefly dynamic transaction accessibility and how to use it to preserve the integrity constraints.

The database integrity is only one aspect of the general database consistency problem. There are other aspects, e.g., transaction consistency problem. The HDDBS model we proposed in this paper is not powerful enough to study the global transaction consistency problem. To do so, we need to incorporate more semantic information, e.g., inter subtransaction value dependency [DE89b], into our model. This is, however, out of the scope of this paper, and will be investigated in the future.

References

[BS88] Y. Breitbart and A. Silberschatz. Multibdatabase update issues. In *Proceeding of the International Conference on Management of Data*, pages 135–142, June 1988.

[DE89a] W. Du and A. Elmagarmid. Quasi serializability: a correctness criterion for global concurrency control in interbase. In *Proceedings of the International Conference on Very Large Data Bases*, Amsterdam, The Netherlands, August 1989.

[DE89b] W. Du and A. Elmagarmid. Supporting value dependency for nested transactions in interbase. Technical Report CSD-TR-885, Purdue University, May 1989.

[DELO89] W. Du, A. Elmagarmid, Y. Leu, and S. Ostermann. Effects of autonomy on global concurrency control in heterogeneous distributed database systems. In *Proceedings of the Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Gaithersburg, MD, October 1989.

[KGM87] B. Kogan and H. Garcia-Molina. Update propagation in bakunin data networks. In *Proceeding of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 13–26, August 1987.