# An Approach to Supporting Semantic Integrity of Federated Databases*

Can Türker

Institut für Technische Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Postfach 4120, D–39016 Magdeburg
E-mail: tuerker@iti.cs.uni-magdeburg.de

**Abstract**

In this paper, a very flexible way to realize global integrity enforcement in federated database systems is proposed. By integrating the active rule paradigm into a federated database framework global integrity constraints which refer to multiple component database systems can be supported. Active rules well-known from active databases can be used as a powerful mechanism for enforcing integrity constraints in case of violations.

**Keywords:** federated database systems, integrity of federated databases, global integrity constraints, ECA-rules, active databases.

## 1 Introduction

Considering the current situation in large enterprises, we discover that there is a growing interest in modern information systems which are able to deal with several pre-existing heterogeneous and partly autonomous data sources. Obviously, enterprises need an integrated access to enterprise-wide available information systems in order to be able to appropriately (re)act on the market under certain time-constrains.

A possible solution for the problem described above is to use federated database systems [SL90, LMR90] which provide mechanisms for a uniform (and transparent) access to originally separated databases. However, often there are a lot of interdependencies between the data stored in these independently developed databases [RSK91, LM93]. The detection of such interdependencies (or conflicts) must be done during the design process of the federated database [BLN86, HM93, KCGS95]. After the integration of several independently developed and possibly heterogeneous databases into one logical database, one might specify additional inter-database integrity constraints, e.g. all projects in the enterprise must have a unique project number.

Enforcement of such global integrity constraints is thus a very important issue of federated database systems. In the literature, e.g. [BGS92, BST92, ZE93], there is lot of work

*This research was partially supported by the German Country Sachsen-Anhalt under FKZ: 1987A/0025 (Federating Heterogeneous Database Systems and Local Data Management Components for Global Integrity Maintenance) and by the ESPRIT Basic Research Working Group No. 8319 ModelAge (A Common Formal **Model** of Cooperating Intelligent **Age**nts).

on the problem of (global) transaction management in federated database systems. Here, especially the topics of ensuring global serializability and global atomicity, respectively, and of handling global deadlocks are intensively discussed.

In this paper, we focus our attention on providing additional mechanisms for global semantic integrity constraint enforcement. For that, we propose an extended active database framework which supports active rules known from the active database field. We motivate the usage of active rules for detecting and signaling events which possibly violates the global integrity. In case of integrity violation, these active rules can be used to initiate appropriate integrity preserving or restoration operations, respectively.

The remainder of this paper is organized as follows: Section 2 briefly recalls the basic notions and architectural issues of federated database systems following the survey paper of Sheth and Larson [SL90]. Thereafter, Section 3 addresses the problem of integrity maintenance in federated databases. Section 4 presents an extended federated database system framework which incorporates the active rule mechanism for enforcing global integrity constraints. Finally, Section 5 concludes by giving an outlook on future work.

# 2 Federated Database Systems

A *federated database system* (FDBS) is a *distributed* system consisting of a collection of cooperating but (partly) *autonomous* and possibly *heterogeneous* component database systems (CDBSs) [SL90]. Thus, an FDBS is characterized by three orthogonal dimensions: distribution, heterogeneity, and autonomy. Besides, federated database systems are categorized as loosely coupled or tightly coupled. In the loosely coupled approach the user (i.e. the application programmer) is responsible for creating and maintaining the federation, whereas in the tightly coupled approach the federation is created and maintained by the administrator(s) of the federated database system (i.e. by the global as well as local administrators).

The architecture of a federated database system contains a federation layer which provides an interface for the *uniform* (and *transparent*) access to originally separated databases (see Fig. 1). This layer incorporates essential database management facilities like (global) transaction and recovery management. In federated database systems there are two kinds of transactions. On the one hand, there are the existing *local transactions* which are directly submitted to the specific component database systems. On the other hand, there are the *global transactions* which may access data from several component databases by using the interface of the federation layer. These global transactions are decomposed by the global transaction management into so-called *global subtransactions* which are submitted to the corresponding component database systems.

A crucial aspect of federated database systems is that the participating component database systems stay independent up to a certain degree. In particular, component database systems retain their interfaces so that existing applications can continue performing their local operations without any modifications. In consequence, these local operations are performed outside the control of the federation layer and, thus, may violate global integrity of the federated database. Therefore, an additional mechanism is required which make it possible to control such "harmful" local operations.

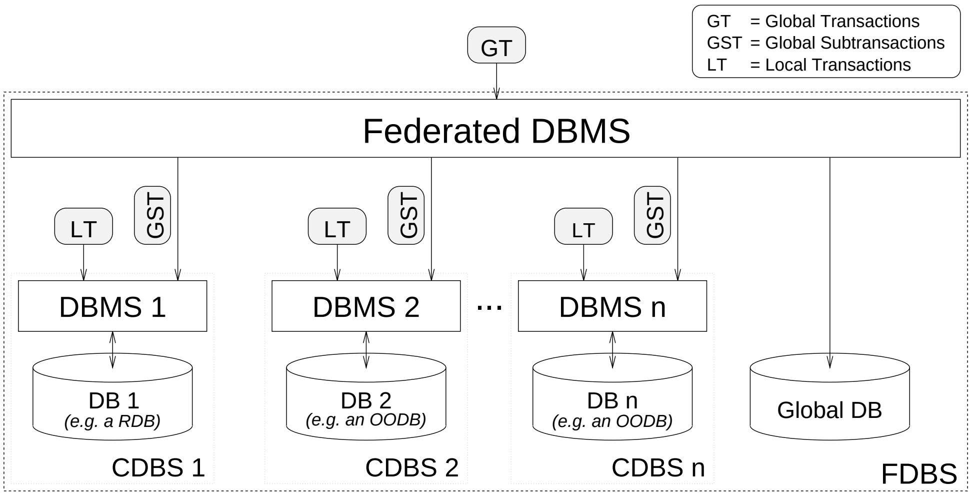


Figure 1: Architecture of federated database systems.

# 3 Integrity of Federated Databases

One of the most important issues of a database management system is to ensure the correctness (i.e. the integrity) of the data managed in the corresponding database(s). In general, two kind of consistency notions are distinguished: the transaction and the database consistency. Whereas the transaction consistency denotes the correctness of interleaved transaction executions in multi-user environment (e.g. conflict serializability as correctness criterion) , the database consistency refers to correctness of the data according to the specified semantical integrity constraints. These integrity constraints restrict the possible database states and database state transitions, respectively, to the valid ones. Thus, a database is called to be consistent if all defined integrity constraints hold.

In centralized database systems, ACID transactions ensure that a database is transferred from one consistent database state to another [BHG87]. For that, a database system provides concurrency, recovery, and integrity control mechanisms. Whereas the concurrency control component guarantees the correct interleaved execution of multiple transactions, the recovery control component secures the database consistency against failures. The integrity control component enforces the semantical integrity constraints, generally at the end of a transaction. In distributed homogeneous database systems the database integrity is ensured by the combination of global synchronization and atomic commitment protocols (e.g. the strict 2PL protocol with 2PC [BHG87, ÖV91]), whereby an integrity control mechanism is often integrated into the commitment protocol.

In comparison to traditional distributed database systems, the consistency enforcement in federated database systems is more difficult. The reason for that is the heterogeneity and autonomy of the component database systems. In the recent years, the problem of transaction management in federated database systems has been discussed in many articles, e.g. [BGS92, BST92, ZE93]. Basically, the focus of these articles was primary on providing mechanisms for ensuring the transaction consistency in federated database systems. In particular, the problems of global serializability, global atomicity and recovery, and of global deadlocks has been addressed. In contrast, the problem of enforcing semantical integrity constraints in a federated database environment has been paid little attention. At the time being, the notion of integrity of federated databases is not clearly defined. Therefore, let us have a look at the integrity constraints existing in database federations.

At the component system level, there are the local integrity constraints which are defined on the local database schemata. Because not all local schemata are exported to a federation, there are local constraints which are not relevant for the integrity of a federated database. At the federation level, there are the global integrity constraints (e.g. inter-database key constraints) which are defined on the federated database schemata. These constraints generally concern multiple local databases. Thus, we may informally define: *A federated database is consistent if all global integrity constraints as well as all integrated (local) integrity constraints hold.*

Basically, three typical kinds of global integrity constraints exist. These are the global key constraints, the global referential integrity constraints and the global aggregate constraints. Global referential integrity constraints are used to describe relationships between two objects of different local databases, whereas global aggregate constraints are used to model constraints on multiple objects in different local databases (e.g. the sum of all projects of all departments must not exceed a certain value). A global key constraint ensure the uniqueness of an object of the federated schema.

For example in a large enterprise where usually each department manages data concerning to their projects, we may specify the condition that each project must be determined unequivocal, for instance, by a unique project number. The specification of such a constraint can be trivial in some cases, whereas the enforcement is generally very difficult. Suppose, we have a federated schema which is an integration of several local schemata. In order to specify a global key attribute, an attribute of the integrated schema must be marked as primary key, for example in an abstract notation as follows:

```
object type Project (
  number: char(10) not null,
  ...
  primary key (number)
);
```

In order to enforce such a global key constraint, all relevant insert and update operations have to be controlled. Consequently, an FDBS has to ensure that no global as well as no local transaction can

- create a new project (object) with an existing project number or
- update the project number to an existing value, respectively.

In consequence, already for a such simple constraints an FDBS has to control a lot of operations. The number of these operations increases if local databases base object-oriented data models which support arbitrary methods for updating attributes of an object.

Intuitively, for the enforcement of global integrity constraints an FDBS requires a integrity control mechanism which allows to detect integrity violations by local as well as global transactions. This implies that a certain reduction of the local autonomy is unavoidable if global integrity enforcement should be supported by an FDBS.

# 4 Towards an Active Federated Database Framework

This sections discusses the idea of realizing global integrity constraints as active rules and sketches a framework of an active federated database system which provides a powerful and flexible platform for global integrity constraint enforcement.

## 4.1 Basic Idea: Using Active Rules for Realizing Global Integrity Constraints

Our basic idea is to adopt the active rule (or ECA-rule) paradigm [Buc94, DHW95] to a federated database framework and to use it as basic mechanism for detecting events which potentially violate the integrity of the federated database. Global integrity constraints and possible *integrity restoration actions* can be informally specified as ECA-rules as follows:

**define rule** *Global_Integrity_Rule*
**on** *event which potentially violates the integrity of the federated database*
**if** *a global condition formulated over the state of the federated database is true*
**do** *global action(s) which restore(s) a consistent federated database state*

With ECA-rules we able to define events which potentially violate the consistency of the federated database and corresponding (re)actions on these events which should restore a consistent database state by aborting the operation/transaction or starting compensating transactions.

## 4.2 Global versus Local Rules

In order be able to efficiently detect potential integrity violations, an FDBS has to support active rules at both levels of an FDBS. At the federation level, *global ECA-rules* are required to detect possible global integrity violations and to initiate restoration actions. By using global ECA-rules an FDBS is able to react on situations which are caused by either global applications or the (global) system clock (e.g. absolute time events [GD94]). However, for global integrity enforcement global ECA-rules alone are not sufficient. Indeed, we may specify global ECA-rules to avoid that global transactions violate the integrity of the federated database. But, global ECA-rules cannot prevent local transactions from performing operations which may have an effect on the global integrity.

Therefore, *local ECA-rules* are needed which may be used to signal the occurrence of a local event which may cause a violation of the global integrity of the federated database. In contrast to global ECA-rules, the only purpose of local ECA-rules is to signal the occurrence of a specific local event to the federation layer. Hence, the format of local E(C)A-rules is as follows:

**define rule** *LocalRule*
**on** *local event*
**do** *trigger the corresponding global rule*

## 4.3 From Global Integrity Constraints to Global and Local ECA-rules

As we have mentioned before, each operation whose execution may run into a inconsistent state of the federated database must be controlled. Because global integrity constraints may be arbitrarily complex, a lot of rules for detecting integrity violating operations has to be generated. Obviously, for the derivation of local and global ECA-rules from global integrity constraints a tool support is required.

For each global integrity constraint a global ECA-rule has to be derived. This rule is a kind of coordination rule whose event part is a composite one referring to the events which may potentially violate the corresponding constraint. Consequently, the number of local rules which trigger the global rule depends on the component databases involved in such a global constraint.

Because local ECA-rules are only used to trigger the corresponding global ECA-rule, local ECA-rules are assumed to have no condition part. Thus, local ECA-rules are actually EA-rules which contain only a notification operation in the action part. Therefore, the mapping process from global to local ECA-rules primarily consists of the decomposition of a composite event in several primitive events, and can be executed completely automatically (by using the metadata of the federated database).

## 4.4 An Extended FDBS Architecture with Active Capabilities

As we have seen in Section 2, traditional federated database systems consist of two levels: the component system level and the federation level. The federation layer just provides a uniform access to all component database by transforming incoming global transactions to local transactions and sending it to the right component system. However, in such a architecture neither global transaction nor integrity control can be supported.

In Figure 2 an extended federated database system architecture is depicted which we will discuss in the following.

The federation layer provides the global interface which is used by the global transactions to access data from multiple component databases. This layer provides full database functionality and moreover serves as a global coordinator which mediates betweens the different data representation forms of the federation layer (common data model) and the component systems (native data models).

The federation layer also incorporates a global transaction management component which provides advanced nested transactions. We suppose to have a transaction model similar to the one presented in [BÖH+92] which supports closed as well as open nested subtransactions, vital and non-vital subtransactions, contingency transactions, and compensating transactions.

All global transactions are subdivided to global subtransactions which may be executed on the local systems. Before these subtransactions can be passed to the mediator of the corresponding component database system, the global transaction management checks whether there are global serializability conflicts. In case there is a conflict the global transaction has to wait, otherwise the required recovery informations are stored (e.g. for compensation) and the subtransactions are directed to the corresponding mediators.

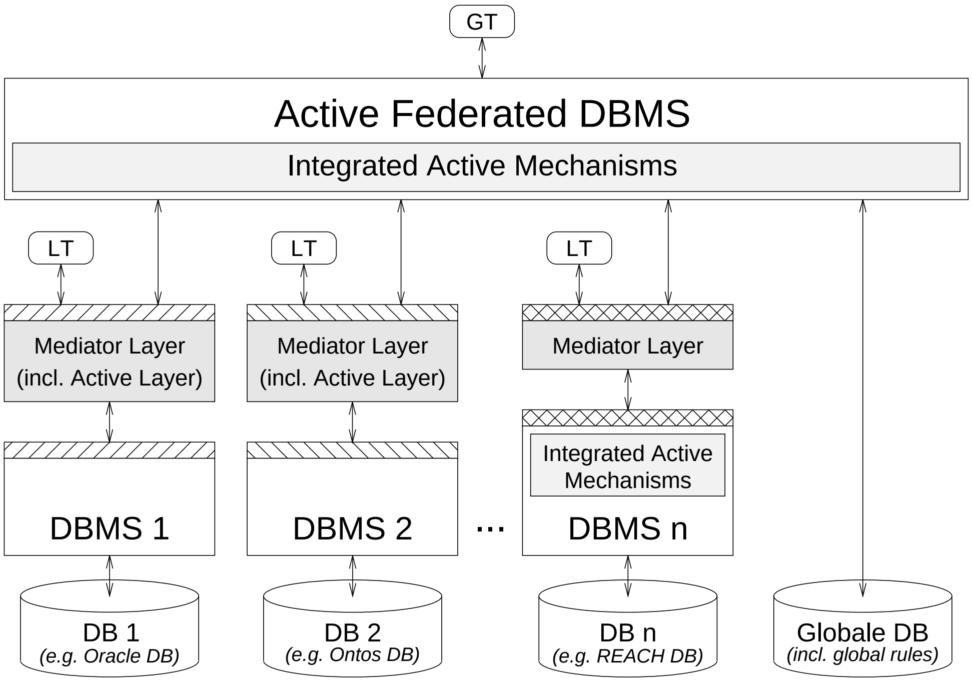


Figure 2: Active federated database framework

### Mediator Layers for Component Database Systems

In order to provide global transaction management which supports global serializability and global atomic commitment, the introduction of an additional layer on top of the components systems is necessary. Indirect conflicts, for instance, cannot be detected by the global transaction management without any informations about the local transactions [BST92, ZE93, SSW95]. Therefore, for each component database system there must a mediator which is used to wrap this local system. So local transactions have to pass this mediator layer in which the global serializability checking routines are integrated. Moreover the mediator can be used to implement an atomic commitment protocol such as two-phase commit.

Because the interface of a mediator corresponds to the original one of the corresponding component database system, the existing local applications may continue without any modifications. All local database calls are passed to the mediator which is then able to analyze the requests. If a request causes a (serializability) conflict it can be rejected, otherwise it may be passed to the original database system interface.

### Integrated Active Mechanisms

In order to support the management and execution of global and local ECA-rules, active mechanisms are incorporated into both levels of an FDBS (cf. Figure 2).

At the federation level, global active mechanisms are offered in form of *global rule managers*. These rule managers are responsible for the management and processing of global ECA-rules. These rules are stored in the global database which also contains all data of the federation layer (e.g. metadata for schema integration). According to the event part, every global ECA-rule is attached to exactly one global rule manager. Especially, a global rule manager is responsible for

- detecting global events which refer to either operations on the global database or absolute time events,
- collecting all incoming messages from the local rule managers and, thus, detecting global composite events,
- evaluating the condition of an ECA-rule after the occurrence of an event, and
- starting the execution of action(s) when the condition is satisfied.

All global active components (i.e. global rule managers) are integrated into the federation layer similar to the REACH approach (cf. [BZBW95]).

At the component system level, active mechanisms are provided in form of *local rule managers*. These rule managers are responsible for the management and processing of local ECA-rules. In particular, a local rule manager detects local events and notifies the corresponding global rule managers in case of event occurrences.

Please notice, if a component database system does not support any active mechanisms, an additional active layer on top this system is required (see the SAMOS approach [GGD94]). As depicted in Figure 2 this layer can be seamlessly integrated into the mediator layer.

## 4.5 Examples for Active Global Integrity Maintenance

In this section, we exemplarily show how global integrity maintenance in an active federated database environment can work. Especially, we present an example how global integrity constraints such as global key constraints can be specified in the proposed active federated database framework.

For that, we assume that we have the following application scenario: An enterprise wants to couple all (or parts of) the database systems of its departments to a global federated database system. Suggest, there are two databases: (1) a relational database DB1 which contains a relation Employee where all data of the employees of the department 'Implementation' are recorded and (2) an object-oriented database which has a class DeptMembers in which all data concerning the employees of the department 'Design' are recorded. Further, let us assume that these local database schemata are integrated into a single federated schema, on which we can define global integrity constraints.

In relational database systems, key constraints are used to model the uniqueness of a real-world entity. Although in object-oriented database systems objects are identified by an internal identifier mechanisms, in general additional key attributes are explicitly used to make it possible for the user to distinguish between different objects.

Key constraints are very important for FDBSs, too. Each object of an FDBS has to be identifiable not only in the CDBS but also in the whole FDBS. In order to guarantee the global uniqueness of an FDBS object, global key constraints may be defined, e.g. each project in the enterprise is identified by a unique project number. In order to realize such a constraint adequately, we can use active mechanisms which are represented in the active federated framework as follows: For each CDBS which is affected by the integrity constraint a local rule manager for supervising manipulations on the corresponding attribute(s) is created. Moreover, at the federation level there is a global rule manager which is triggered by these local rule managers when such a manipulation operation is requested locally. In this case, the global rule manager has to check whether this operation violates the global integrity. In case of a violation the local operation has to be rejected. For that, a global ECA-rule can look like as follows:

**define rule** *GlobalKeyConstraintRule*
**on** *manipulation of key attribute(s)* **or** *creation of a new object*
**if** *global key constraint is violated*
**do** *abort the corresponding transaction*

# 5 Conclusions and Outlook

In this paper we addressed the problem of global integrity constraints in federated databases and sketched a possible advanced federated database system architecture which supports the enforcement of such interdatabase constraints. In combination with a multidatabase transaction management which prevents concurrent global transactions from seeing inconsistent states, the active rule mechanism provides a very flexible and powerful basis for realizing global integrity constraints in database federations.

However, there are a lot of problems which have to be investigated in future work. For instance, we have to fix a rule definition language which is tailored to the requirements of global integrity constraint enforcement. Furthermore, we have to develop appropriate methods which allows a semi-automatically derivation of global and local rules from specified integrity constraints. Another very important issue concerns the synchronization of local and global integrity checking. For that, we have to provide formal basis on which synchronization problems can be worked out.

# References

[BGS92] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Overview of Multidatabase Transaction Management. *The VLDB Journal*, 1(2):181–240, October 1992.

[BHG87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Reading, MA, 1987.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

[BÖH+92] A. Buchmann, M. T. Özsu, M. Hornick, D. Georgakopoulos, and F. Manola. A Transaction Model for Active Distributed Object Systems. In A. K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 123–151. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[BST92] Y. Breitbart, A. Silberschatz, and G. R. Thompson. Transaction Management Issues in a Failure-Prone Multidatabase System Environment. *The VLDB Journal*, 1(1):1–39, July 1992.

[Buc94] A. P. Buchmann. Active Object Systems. In A. Dogac, M. T. Öszu, A. Biliris, and T. Selis, editors, *Advances on Object-Oriented Database Systems*, pages 201–224. Nato ASI Series, Springer-Verlag, Berlin, 1994.

[BZBW95] A. P. Buchmann, J. Zimmermann, J. A. Blakeley, and D. L. Wells. Building an Integrated Active OODBMS: Requirements, Architecture, and Design Decisions. In P. S. Yu and A. L. P. Chen, editors, *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95), Taipei, Taiwan*, pages 117–128. IEEE Computer Society Press, March 1995.

[DHW95] U. Dayal, E. Hanson, and J. Widom. Active Database Systems. In W. Kim, editor, *Modern Database Systems*, pages 434–456. ACM Press, New York, 1995.

[GD94] S. Gatziu and K. R. Dittrich. Events in an Active Object-Oriented Database System. In N. W. Paton and M. H. Williams, editors, *Proc. of the 1st Int. Workshop on Rules in Database Systems (RIDS'93), Edinburgh, Scotland, August 1993*, pages 23–39. Workshops in Computing, Springer-Verlag, 1994.

[GGD94] S. Gatziu, A. Geppert, and K. R. Dittrich. SAMOS: an Active Object-Oriented Database System. Technical Report 94.16, University of Zurich, 1994.

[HM93] S. Hammer and D. McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous Database Systems. *Int. Journal of Intelligent and Cooperating Information Systems*, 1(2):51–83, 1993.

[KCGS95] W. Kim, I. Choi, S. Gala, and M. Scheevel. On Resolving Schematic Heterogeneity in Multidatabase Systems. In W. Kim, editor, *Modern Database Systems*, pages 521–550. ACM Press, New York, 1995.

[LM93] Q. Li and D. McLeod. Managing Interdependencies among Objects in Federated Databases. In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *Interoperable Database Systems (DS-5), Proc. of the IFIP WG 2.6 Database Semantics Conf., Lorne, Victoria, Australia, November, 1992*, pages 331–347. North-Holland, Amsterdam, 1993.

[LMR90] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[ÖV91] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, Eaglewood Cliffs, New Jersey, 1991.

[RSK91] M. Rusinkiewicz, A. Sheth, and G. Karabatis. Specifying Interdatabase Dependencies in a Multidatabase Environment. *IEEE Computer*, 24(12):46–53, December 1991.

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SSW95] W. Schaad, H.-J. Schek, and G. Weikum. Implementation and Performance of Multi-level Transaction Management in a Multidatabase Environment. In O. Bukhres, T. Özsu, and M.-C. Shan, editors, *Proc. of the 5th Int. Workshop on Research Issues on Data Engineering: Distributed Object Management (RIDE-DOM'95), Taipei, Taiwan*, pages 108–115. IEEE Computer Society Press, Los Alamitos, CA, March 1995.

[ZE93] A. Zhang and A. K. Elmagarmid. A Theory of Global Concurrency Control in Multidatabase Systems. *The VLDB Journal*, 2(3):331–360, July 1993.