Towards Maintaining Integrity of Federated Databases*

Can Türker and Stefan Conrad

University of Magdeburg
Institute of Technical Information Systems
P.O. Box 4120, D–39016 Magdeburg, Germany
Email: {tuerker|conrad}@iti.cs.uni-magdeburg.de
Phone: ++49/391/67-{12994|18066} Fax: ++49/391/67-12020

Abstract

Supporting integrity constraints is essential for database systems. In consequence, global integrity constraints must be dealt with by federated database systems. Although a lot of work has been done on transaction management in multidatabase systems, the treatment of semantic integrity constraints (e.g. uniqueness conditions), especially detecting (potential) violations and specifying necessary reactions, is often omitted.

In this paper, we propose a very flexible way to realize global integrity maintenance in federated database systems. We aim at integrating the active rule paradigm into a federated database framework. Thereby, integrity constraints involving multiple component database systems can easily be supported. Active rules provide a powerful mechanism for communication and co-operation between heterogeneous database systems. In addition, they are to be used for specifying enforcement of constraints in case of violation. For enforcing constraints the existing local mechanisms (e.g. triggers) of the component database systems can be employed. In case this is not possible, an additional layer has to be introduced which realizes the active rule mechanism for the component database system.

Keywords: active integrity maintenance, global integrity constraints, federated database systems, ECA-rules.

1 Introduction

Considering the current needs of enterprises in federating several pre-existing database systems to obtain one enterprise-wide available information system, it clearly comes to light that global integrity constraints are necessary. A federated database system consists of a number of possibly heterogeneous database systems which are considered to be (partly) autonomous [17]. Obviously, there are interdependencies between data stored in originally separate databases [12, 11]. For instance, mutual consistency must be ensured by the federated system. Therefore, global integrity maintenance becomes an essential property of federated database systems.

The approach we describe in this paper is based on mechanisms as they are used for integrity enforcement in active database systems [8]. From our point of view, active rules (ECA-rules) are a promising principle not only for integrity enforcement in single, centralized database systems, but also for federated heterogeneous systems. This is due to the fact that the active rule paradigm is in principle system-independent. The active rule mechanism can be considered as a communication mechanism between the component database systems and the federation layer. Therefore, it is rather straightforward to use active rules on the global (federation) level of the system to specify integrity constraints and actions which have to be executed in case of potential integrity violations.

In this paper, we focus on the active rule mechanisms. The next step to consider is the co-operation of active rule mechanisms and transaction management in federated database systems. At first glance the integration of our approach into transaction models for multidatabase systems, e.g. [4, 13], seems to be nearly problem-free, but we have to work out this in more detail.

The remainder of this paper is organized as follows: In Section 2, we present our approach of integrating active rules mechanisms into the federated database architecture for providing an adequate and efficient mechanism for detecting potential integrity violations and triggering corresponding actions. In Section 3 we demonstrate by example how various global integrity constraints can

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

be maintained in an active federated database environment. Afterwards, related work is briefly surveyed in Section 4. Finally, we conclude by giving an outlook on future work.

2 Towards an Active Federated Database Framework

In this section, we present an active federated database framework providing a powerful and flexible platform for global integrity constraint enforcement in federated database systems. Thereby, we pursue the approach to integrate active mechanisms (ECA-rules) into a federated database system for supporting global constraint management. Besides, a main idea is to include existing integrity mechanisms of the component database systems (CDBSs).

2.1 Foundations

In the last decade, active database systems (see for a survey [8]) have been proposed as a new data management paradigm to capture more real-world semantics already in the database rather than in applications programs. Active database systems are extended full-fledged database systems which are able to monitor predefined situations (situations of interest) and to react to them under given time constraints with likewise predefined action. Such (re)active behavior is generally expressed by the so-called *event-condition-action rules (or ECA-rules)* which define *what to do* if a certain situation occurs in the database. ECA-rules can be used as uniform mechanism for integrity constraint enforcement, derived data maintenance, version control, and several other database features.

Our goal is to adopt the ECA-rule paradigm to a federated database framework and to use it as basic mechanism for detecting events, which potentially violate the integrity of the federated database as well as for communication between the federation and the component database level.

In general, the integrity of federated data is very difficult to maintain, because all data, locally as well as globally created database objects, has to fulfill certain consistency requirements. These requirements are usually expressed by *global integrity constraints*. A federated database is called to be consistent if all global integrity constraints as well as all existing local integrity constraints hold. In the literatur, for instance [6], there are several ways proposed how integrity of a database can be maintained:

1. rolling back any (global as well as local) transaction that produces an inconsistent state,
2. disallowing changes that may cause an inconsistent state,
3. forcing to use only predefined, secure transactions, or
4. using rules to restore a consistent state.

By using ECA-rules we can specify global integrity constraints and possible *consistency restoration actions* 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*

So we are 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 (trans)actions.

In the following subsection we present an enhanced federated database system architecture which integrates the active rule mechanism and thus provides a suitable platform for integrity maintenance and integrity enforcement, respectively, in federated environments.

2.2 Integration of Active Mechanisms into an FDBS

A federated database system consists of two levels: the component system level and the federation level. The integration of active mechanisms into such an environment can be done in the following way (cf. Figure 1).

At the federation level, global active mechanisms are offered in form of *global ECA-rules*. These ECA-rules are stored in the global database which also contains all data of the federation layer (e.g. meta information for schema integration). According to the event part, every global ECA-rule is attached to a *global rule manager* which is responsible for the management and processing of this ECA-rule. These active components are integrated into the federation layer similar to the REACH approach (cf. [5]).

By using global ECA-rules we are able to specify (re)actions on situations which are caused by either global applications/users or the (global) system clock

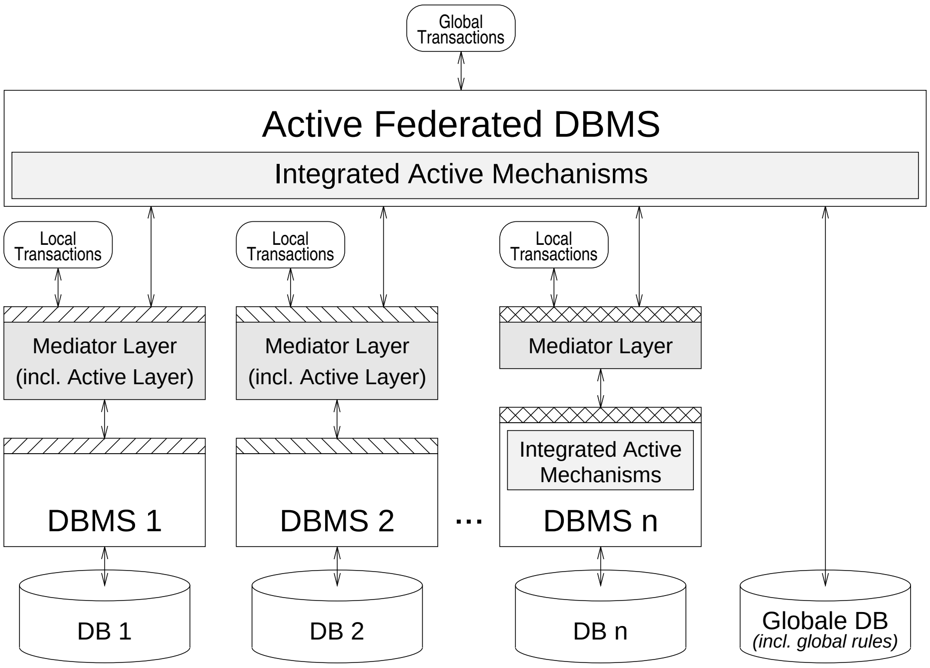

Figure 1: Active federated database framework

(e.g. absolute time events). For global integrity enforcement global ECA-rules are not sufficient. Indeed, we may specify global ECA-rules in order to avoid that global applications violate the integrity of the federated database. But, global ECA-rules alone cannot prevent local applications from performing operations which may have an effect on the global integrity.

Therefore, we introduce the notion of *local ECA-rules* and *local rule managers*, respectively, which reside in a mediator layer on top of the local CDBSs. Local ECA-rules are used to specify (re)actions which are performed upon the occurrence of a local event. Like global ECA-rules, local ECA-rules are managed and processed by a (local) rule manager. A local rule manager detects a local event in the CDBS and informs the appropriate global rule manager about the event occurrence. In contrast to global ECA-rules, local ECA-rules (as the name suggests) are stored locally at the component system level. In our approach, the only purpose of these 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 *inform the corresponding global rule manager*

In the following subsection, we discuss possible representations of local ECA-rules in federated environments.

2.3 Representation of Local ECA-Rules

A general question is how can we adequately represent local ECA-rules as well as their local rule managers inside an FDBS? Obviously we need some active mechanisms for that at the CDBS level. In principle, there are two possible scenarios which we discuss in the following:

1. **Using existing active capabilities of the CDBS:** If the underlying CDBS supports some active mechanisms (e.g. triggers), we might use them to signal the occurrence of a local event to the corresponding global rule manager.

   Today, most of the commercial relational database systems support some kinds of trigger mechanisms which are already very powerful. Triggers, which can be seen as a special kind of active mechanisms, can be used to signal the request of a SQL operation to the federation layer. The term trigger is often used in the context of relational database systems where triggers may be defined on database operations like insert, update, delete, or select:

   In contrast, if the underlying CDBS is an active one which supports ECA-rules the case is trivial. Here, the existing local ECA-rule mechanism can be used as described in the previous subsection.

2. **Providing an additional layer with active mechanisms:** In case a CDBS does not support

any active mechanism, we have to provide an active layer á la SAMOS [9] for that CDBS (see DBMS1 and DBMS2 in Figure 1). So, we are able to define and handle local ECA-rules in arbitrary CDBSs.

In general, a technique which is called method wrapping (i.e. putting additional code into an application program) can be applied to equip a database system with active capabilities. Method wrapping is successfully used in active object-oriented database systems (see for instance [9, 5]).

In case of relational database systems we may use a similar technique. When a SQL-operation (e.g. an update) is sent to the system it has to pass the active layer. Here, we are able to detect events before and after passing the operation to the original (passive) system, respectively.

2.4 Derivation of Local ECA-Rules from Global ECA-Rules

Another very important aspect concerns the derivation of local ECA-rules from global ECA-rules. A global ECA-rule may base on a composite event which consists of several local events (in different databases), for example

define rule *RuleX*
on *DB1::e1(p1)* **or** *DB2::e2(p2)*
if ...
do ...

Such a global ECA-rule has to be decomposed into several local ECA-rules, i.e. here two local ECA-rules have to be generated from the rule *RuleX*.

define rule *Local_IntegrityRuleX1*
on *DB1::e1(p1)*
do *GDB::notify(RuleX, DB1::e1(p1))*

define rule *Local_IntegrityRuleX2*
on *DB2::e2(p2)*
do *GDB::notify(RuleX, DB2::e2(p2))*

Because local ECA-rules are only used to trigger the corresponding global ECA-rule (here: *RuleX*), 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).

In conclusion, the combination of global and local ECA-rules allows to realize global integrity constraints which must be satisfied by global as well as local applications. These rules are managed and processed by corresponding rule managers. On the one hand, there are the local rule managers which are responsible for

- detecting local events in the CDBSs, and
- notifying the responsible global rule manager about the occurrence of that event.

On the other hand, the global rule managers are 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.

As we have seen in this section, applying active mechanisms for integrity enforcement in (active) federated database environments is very promising. In the following section, we present some examples to underline this statement.

3 Examples for Global Integrity Maintenance in an Active FDBS

In this section, we exemplarily show how various global integrity constraints can be maintained and enforced, respectively, in an active federated database environment.

Assume, there exists the following application scenario: A large enterprise consists of several departments which manage their independently developed databases separately by heterogeneous database systems. The enterprise wants to couple all (or parts of) these standalone systems to a global federated database system. Let us further suppose that

- CDBS1 is an RDBS which supports simple trigger, mechanisms,
- CDBS2 is an OODBS with no active capabilities, and
- CDBS3 is an AOODBS which provide ECA-rule mechanisms.

Moreover, let us assume that there exist several heterogeneous databases which are managed by the federated CDBSs:

- CDBS1 manages a relational database which contains a relation `Employee` (as depicted below), where all relevant data of the employees of the enterprise are recorded.

```
relation Employee
(
  name   : char(20) not null,
  ssn    : int not null,
  address: varchar(80),
  salary : int,
  dept   : string,
  primary key (name, ssn)
);
```

- CDBS2 manages information about the departments of the enterprise.

```
interface Department
{
   extent  departments;
   string  name;
   integer budget;
   integer members;
   string   head;
};
```

- CDBS3 contains an (active) object-oriented database which belongs to the department "DatabaseResearch" and contains data describing the projects of that department and the researchers involved in these projects.

```
interface Researcher
{
   extent              researchers;
   string              name;
   integer             ssn;
   set<ref<Project>>   worksIn
                       inverse members;
};

interface Project
{
   extent                 researchers;
   integer[4]             id;
   string                 name;
   integer[10]            budget;
   ref<Researcher>        manager;
   set<ref<Researcher>>   members
                          inverse worksIn;
};
```

We expect all these local database schemas to be integrated adequately into a(n) (object-oriented) federated schema, on which we now can define global integrity constraints. For that, we have chosen various kinds of integrity constraints to elucidate how active integrity maintenance can work in an active federated database environment as proposed in the previous section.

1. **Attribute constraints:** Attribute constraints restrict the values an attribute can have. Such kinds of constraints are represented, for instance, in relational database systems by the well-known **check** or **not null** clauses. In object-oriented database systems, in contrast, attribute constraints are generally realized in methods which are accessing and manipulating the corresponding attribute.

   In our sample scenario, we may have an additional global attribute constraint saying that no project in the enterprise is allowed to exceed a given budget. Such kinds of constraints can easily be enforced by existing local integrity mechanisms, e.g. by an additional **check** clause, if schema modification is allowed in the corresponding CDBS. Otherwise, we may realize it by ECA-rules as proposed in the previous section.

2. **Key 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, often 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* (cf. [14]). In order to guarantee the global uniqueness of an FDBS object, global key constraints may be defined, e.g. each project in the enterprise must have an 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 manipulation operations 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 *reject the corresponding transaction*

Please note that there are several problems concerning object identification in federated database systems, which have to be considered when enforcing key constraints. For example, we may have replicas or same-objects of one and the same real-world entity in different databases. In such cases, updates of key attributes must be propagated to the primary copy or to all other same-objects of that entity, respectively.

In our example, we may constrain that all researchers must be employees of the enterprise. In order to realize such a constraint, the operations "create a new researcher object" and "change key attribute(s) of a researcher object" has to be monitored. When they occur, it has to be checked whether this global integrity constraint is violated:

define rule *ResearcherMustBeAEmployeeRule*
on *manipulation of the attributes* `Researcher::name` *or* `Researcher::ssn` **or** *creation of a* `Researcher` *object*
if *corresponding* `Employee` *object does not exist*
do *reject the corresponding transaction* **or** *insert relevant data of* `Researcher` *to* `Employee`

3. **Aggregate constraints:** Generally, aggregate constraints depend on several attributes of one object as well as of multiple objects. In our example, we may formulate the constraint that the sum of the budgets of all projects of the enterprise must not exceed a certain amount in the following way:

define rule *GlobalBudgetRule*
on *any* `Department::budget` *manipulation operation*
if *sum of budgets of all* `Departments` *exceeds a given amount*
do *reject the corresponding transaction*

4. **Referential integrity constraints:** In traditional database systems, referential integrity constraints are used to specify existence dependencies between two database objects, i.e. they guarantee that a database object refers to existing objects.

In our sample scenario, we may define the constraint that if an employee leaves the enterprise he/she must also leave all projects he/she is participated in:

define rule *GlobalReferentialIntegrityRule*
on *deletion of an* `Employee`
if *this* `Employee` *is involved in at least one* `Project`
do *remove this* `Employee` *from all corresponding* `Projects`

Please note that by using ECA-rules we are able to realize arbitrary constraints. Especially, we can specify temporal constraints, e.g. in order to model admissible traces of evolution of the federated database. For instance, in our scenario we may specify that the enterprise forbids to start more than five new projects per year in the following way:

define rule *GlobalTemporalConstraintRule*
on *start of new project*
if *already five projects have been started in the same year*
do *reject transaction*

4 Related Work

Although there is a wide spectrum of work in the area of federated database systems, only few approaches deal with the problem of global integrity constraints and of integrity maintenance for federated database systems. Nevertheless, the necessity of global integrity constraints and thus of global integrity maintenance has been recognized by several authors.

[12, 16] address the problem of interdependencies and the effect they have on applications manipulating interdependent data. From their point of view, integrity constraints have to be separated from application programs in order to facilitate the maintenance of integrity constraints. Hence, they propose a model which allows the declarative specification of interdependencies using so-called *dependency descriptors*, which contain not only interdependency constraints but also consistency requirements and restoration procedures. These dependency descriptors are then used to generate so-called *polytransactions* which maintain the mutual consistency among interrelated databases.

In [15] the concept of semantic proximity (degree of semantic similarity) is defined and a semantic taxonomy (semantically equivalent, semantically related, etc.) based on this concept is provided. Further, various schematic and data conflict (e.g. conflicts caused by different naming, scaling, precision of an attribute or object) as well as several kinds of data inconsistencies are enumerated and classified.

A further discussion on interdependencies in federated database systems can be found in [11]. In this paper four kinds of generic interdependency relationships, named existence, structural, behavioral, and value dependencies, are introduced and scenarios for all possible combinations of these interdependencies are described. Moreover, an object-oriented database framework, called ODF, is proposed for managing such interdependencies. But update propagation and federated transaction management issues are not suggested in this paper.

In [4] a transaction model for active distributed object systems, called the DOM transaction model, is presented. This model contains several kinds of transactions, e.g. nested, vital and non-vital, compensating, and contingency transactions, which can also be used as basis for integrity maintenance and enforcement in federated database systems. Other authors, for instance [13], propose similar transaction models which base on open nested transactions and thus allow "early" commits of subtransactions. An analysis of special problems occurring in transaction management in multi-database systems is given in [3].

In addition to transaction models for multi-database systems, a number of special techniques for enforcing integrity constraints have recently be proposed. For instance, the *Demarcation Protocol* [2] and the *Reconfiguration Protocol* [1] are based on the idea that a global integrity constraints can be guaranteed by different configurations of local integrity contraints. In case a local database violates its local constraint without violating the original global constraint, another configuration of local constraints is looked for such that neither a local constraint nor the global one is violated anymore. However, this procedure can only be realized for certain kinds of integrity constraints. Some authors like [10] present local verification techniques which can help to avoid unnecessary communication between the local database systems and the federation layer. Furthermore, there are approaches (e.g. [7]) which are tailored for autonomous distributed databases and, therefore, provide a weaker notion of constraint enforcement.

5 Conclusions

Our proposal for integrity enforcement in a federated, heterogeneous framework is based on the idea to bring active mechanisms into federated database systems. We have shown in which way active rules can be applied in FDBSs for global constraint maintenance. Active rules in federated database systems as we proposed here are an important basic mechanism for global integrity maintenance. Based on this mechanism an appropriate transaction management is needed which allows the co-existence of local and global (federated) transactions. Among the existing proposals for transaction management in multi-database systems there are several candidates for which a combination with our approach seems to be not only possible but also desirable. Such an integrated approach will be a major part of our future work. Nevertheless, it should be clear that preserving local autonomy and enforcing global constraints represent two contrary requirements.

In addition, there are several other questions and problems we are interested in. One problem is the (automated) derivation of local ECA-rules from global integrity constraints. Here, different classes of integrity constraints being relevant for federated databases have to be investigated. A very special and technical problem is the decomposition of global events into local events which themselves can be composed events as well. A first idea could be to restrict the design of global integrity constraints in a certain way such that the decomposition becomes manageable.

Acknowledgements: We would like to thank our colleagues Gunter Saake, Ingo Schmitt, Michael Höding, Kerstin Schwarz, and Andreas Christiansen for the fruitful discussions.

References

[1] G. Alonso and A. El Abbadi. Integrating Constraint Management and Concurrency Control in Distributed Databases. *Bulletin of the IEEE Technical Committee on Data Engineering*, 17(2):18–22, June 1994.

[2] D. Barbará-Millá and H. Garcia-Molina. The Demarcation Protocol: A Technique for Maintaining Constraints in Distributed Database Systems. *The VLDB Journal*, 3(3):325–353, July 1994.

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

[4] 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.

[5] 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, March 1995*, pages 117–128, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[6] S. Ceri and J. Widom. Deriving Production Rules for Constraint Management. In D. McLeod, R. Sacks-Davis, and H.-J. Schek, editors, *Proc. of the 16th Int. Conf. on Very Large Data Bases (VLDB'90), Brisbane, Australia*, pages 566–577, Morgan Kaufmann Publishers, Palo Alto, CA, August 1990.

[7] S. S. Chawathe, H. Garcia-Molina, and J. Widom. Flexible Constraint Management for Autonomous Distributed Databases. *Bulletin of the IEEE Technical Committee on Data Engineering*, 17(2):23–27, June 1994.

[8] K. R. Dittrich, S. Gatziu, and A. Geppert. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. In T. Sellis, editor, *Rules in Database Systems, Proc. of the 2nd Int. Workshop, RIDS'95, Athens, Greece, September 1995*, pages 3–20, Lecture Notes in Computer Science, Vol. 985, Springer-Verlag, Berlin, 1995.

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

[10] A. Gupta and J. Widom. Local Verification of Global Integrity Constraints in Distributed Databases. In P. Buneman and S. Jajodia, editors, *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C.*, pages 49–58, Vol. 22, No. 2, ACM Press, June 1993.

[11] 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.

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

[13] 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, *RIDE-DOM'95, Proc. of the 5th Int. Workshop on Research Issues in Data Engineering: Distributed Object Management, March 6–7, 1995, Taipei, Taiwan*, pages 108–115, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[14] I. Schmitt and G. Saake. Managing Object Identity in Federated Database Systems. In M. Papazoglou, editor, *OOER'95: Object-Oriented and Entity-Relationship Modeling, Proc. of the 14th Int. Conf., Gold Coast, Australia*, pages 400–411, Lecture Notes in Computer Science, Vol. 1021, Springer-Verlag, Berlin, December 1995.

[15] A. Sheth and V. Kashyap. So Far (Schematically) yet So Near (Semantically). 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 283–312, North-Holland, Amsterdam, 1993.

[16] A. Sheth, M. Rusinkiewicz, and G Karabatis. Using Polytransaction to Manage Interdependent Data. In A. K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 555–581, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[17] 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.