Published as: Preprint Nr. 9, Fakultät für Informatik, Universität Magdeburg, November 1995.

Active Integrity Maintenance in Federated Database Systems [1]

Stefan Conrad Can Türker

Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D–39016 Magdeburg
Germany

Email: {conrad|tuerker}@iti.cs.uni-magdeburg.de

Tel.: ++49-391-67-{18066|12994}
Fax: ++49-391-67-12020

October 1995

[1]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 Country Sachsen-Anhalt under FKZ: 1987A/0025 ("Föderierung heterogener Datenbanksysteme und lokaler Datenhaltungskomponenten zur systemübergreifenden Integritätssicherung").

Abstract

We propose the active database paradigm as a suitable approach to realize global integrity maintenance in tightly coupled federated database systems. It has come to light that there is a need for global integrity constraints in federated database systems in order to deal with for instance replicated data and inter-database dependencies. However, only few, mostly limited approaches towards global integrity maintenance have been proposed for different kinds of multi-database systems. Our proposal for tightly coupled federated database systems is more general by integrating active rule mechanisms. Thereby, a very flexible way of expressing integrity constraints involving multiple component database systems is provided. Furthermore, active rules are a powerful mechanism for specifying enforcement of constraints in case of violation. For enforcing constraints the local mechanisms 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,
active databases.

Acknowledgements

We are especially grateful to our colleagues Gunter Saake, Ingo Schmitt, Jan Kusch and Michael Höding for fruitful discussions and useful remarks on preliminary versions of this report.

Contents

Chapter 1

Introduction

Following the needs of present and future database applications, interoperability of database systems becomes more and more important. Today, nearly every enterprise has to face the situation that different database systems are used in its departments. Problems caused by this situation are redundant data, inter-database dependencies and data-exchange between different databases. For each database system there often exists a large number of applications which have to be preserved. On the other hand, new multi-database applications must be supported, for instance for realizing workflow systems.

The idea of *federated database systems* has been developed to bring together these requirements (cf. [SL90]). A federated database system consists of a number of possibly heterogeneous database systems which are considered to be autonomous. There are local applications which are working on one of the local database systems without knowing about the other ones. In addition, there can be global applications which access the federated database system through a global interface without knowing by which local database system the corresponding data is managed.

A major aspect in such environments is the consistency of federated data. Although the federation of autonomous (heterogeneous database) systems is discussed in many publications (e.g. [SL90, LMR90, BHP92, RS94, PGW95]), at the time being there are only partial solutions to global integrity maintenance in federated database systems. On the one hand, global integrity maintenance seems to contradict the autonomy of the local database systems involved. On the other hand, there is a clear demand for having global integrity constraints which are needed for instance to maintain the consistency of replicated data and inter-database dependencies. Therefore, an approach for dealing with global integrity constraints in federated database systems is required respecting as much as possible the autonomy of the local database systems.

For classical, centralized database systems there are several well-known ways of representing integrity constraints. First, some constraints can directly be expressed by concepts of the underlying data model. Thereby, these constraints become inherent ones which are directly enforced by the database system. Second, we can enrich the database schema with declarative constraints (e.g. using predicate logic). Here, the enforcement of constraints is unclear. One way is to offer only pre-defined transactions which guarantee the satisfaction of the constraints. Another way is to modify all user transactions such that the constraints are always respected. And third, we can use trigger mechanisms (or active rules) as far as they are provided by the database system.

The approach we describe in this paper is based on mechanisms as they are used for integrity enforcement in active database systems (e.g. SAMOS [GGD94], REACH [BZBW95], Sentinel [CKTB95]). 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.

A main idea of our proposal is that we want to use as much as possible existing local mechanisms for integrity maintenance. For that, an appropriate mapping between the global and the local mechanisms is needed. Some commercial database systems, e.g. several relational database systems like SYBASE, already provide triggers which may be considered as a subset of active mechanisms. Thereby, an efficient realization seems to be possible by simply using these existing mechanisms. Other (commercial) database systems only offer very restricted mechanisms for integrity enforcement. Here, we aim at an adequate integration. Furthermore, there are systems which do not support any kind of integrity maintenance at all. For these systems an additional layer has to be introduced into the architecture of a federated database system. We have to emphasize that we are considering tightly coupled federated database systems in this paper. This is because we believe that tightly coupled federated database systems are better suited for global constraint enforcement than loosely coupled ones in which global integrity constraints do not play a major part.

The remainder of this paper is organized as follows: We briefly survey the concepts and notions of active database systems in Chapter 2. Chapter 3 recalls the architecture of federated database systems and briefly presents recent approaches to integrity maintenance in multi-database systems. In the main part of this paper (Chapter 4), we propose the application of active mechanisms for integrity enforcement in a federated system. In particular, we show how active mechanisms can be integrated into federated database systems. We illustrate our approach in Section 4.3 by presenting some examples for specifying global integrity constraints by means of ECA-rules. Finally, we conclude by giving an outlook on future work.

Chapter 2

Notions and Issues of Active Database Systems

Recently, *active database systems* (ADBSs) have become a very important research field (see for example [Cha89, Buc94, Gat94, DHW95]). Initially, active database systems have been proposed as a new data management paradigm to capture more real-world semantics already in the database rather than in applications programs. In fact, active database systems seem to provide a powerful and flexible platform, especially, for application domains in which timely or time-constrained response (which cannot be foreseen in advance) to critical situations is required. Examples for such kinds of applications are air-traffic control, real-time plant control, network management, or workflow management. Common to all these application domains is the demand for handling dynamically changing behavior and environment in an efficient and appropriate way.

Basically, an ADBS is characterized as an extended full-fledged database system which supports *(re)active behavior* beside the conventional database features like transaction management, concurrency control or recovery control. In other words, an ADBS is an extended conventional database system which has the capability 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.

During the last decade, a number of research prototypes for active database systems have been built. Examples for relational database systems with active capabilities are POSTGRES [SHH87, SHP88], Ariel [Han89, Han92], and Starburst [WF90, WCL91]. Examples for active object-oriented database systems are HiPAC [DBB$^+$88, CBB$^+$89], ODE [GJ91, GJS92b], REACH [BBKZ94, BZBW95], SAMOS [GD92, GD94b], and Sentinel [CM93, CKAK94, CKTB95].

In the following sections, we survey the essential features of active database systems. We begin with recalling the ECA-rule concept, thereafter we present the common ingredients of rule definition, rule management and rule execution models, and finally we conclude this chapter by very briefly presenting two well-known architecture approaches of active database systems.

2.1 The ECA-Rule Concept

Today, the ECA-rule concept as introduced in the HiPAC project [DBB+88] is a widely accepted means for describing active capabilities of active database systems. An ECA-rule consists of three parts: an event, a condition, and an action.

on *<event>*
if *<condition>*
do *<action>*

The event part of an ECA-rule specifies *when* the rule is to be *triggered (or fired)*, the *condition* determines *if* the action shall be executed, and the *action* defines *what to do* in this case. An ECA-rule can be degraded to an *EA-rule* if the condition part is missing. In that case the action is executed whenever the corresponding event occurs.

2.2 Rule Definition

An active database system must provide a rule definition language as means for specifying (re)active behavior in forms of ECA-rules. In the last years, several languages were proposed for *active relational database systems (ARDBSs)* as well as for *active object-oriented database systems (AOODBSs)*.

Event Specification: An event is related to a given point in time. In the literature, e.g. [GJS92a, CM93, GD94b, BBKZ92], the following event types[1] are identified:

- *database event*: insertion, deletion, update, and selection of a database object.
- *transaction event*: begin, commit, and abort of a transaction.
- *method call event*: call of an arbitrary method.
- *temporal event*: absolute, relative, and periodic time events.
- *external event*: arbitrary event induced by external devices, users, etc.
- *composite event*: complex event constructed by operators of an event algebra.

[1] From now on, we simply talk about events whenever the distinction to event instances which occur at run-time is clear from the context.

Events are classified into primitive and composite events. Primitive events in general refer to database or transaction operations, or to arbitrary method calls (in case of an AOODBS). Besides, absolute time events [DG93] and external (or abstract) events are also considered as primitive events. In general, most of the rule definition languages support additional *before* and *after* specifiers to refer to the begin or the end of the execution of an operation (or method). Composite events [GJS92b, GD94a, CKAK94], on the other hand, are events which are composed by several "smaller" events using event operators. In the following, we recall the generally accepted event operators[2].

- *Disjunction*: <*event-1*> **or** <*event-2*>
- *Conjunction*: <*event-1*> **and** <*event-2*>
- *Sequence*: <*event-1*> **then** <*event-2*>
- *Closure*: **all** <*event-2*> **in** <*event-1*> , <*event-3*>
- *History*: <*num*> **times** <*event-2*> **in** <*event-1*> , <*event-3*>

Of course, rule definition languages vary in the complexity of specifiable events because of different data models. Whereas AOODBSs support in general most of the events mentioned above, ARDBSs provide only a subset of these events, for instance only database events. Moreover, events in AOODBSs can be parameterized. Thus event parameters can be referred in the condition and action part of an ECA-rule.

Condition Specification: Whereas AOODBSs support arbitrary predicates and queries over the database state as condition, ARDBSs normally allow only to formulate predicates. In case of a predicate, the condition is satisfied if the predicate is true. In the other case, the condition is satisfied if the result of the query is non-empty.

Action Specification: The action part of an ECA-rule specifies the operations to be performed when the ECA-rule is triggered and its condition is satisfied. Whereas in AOODBSs arbitrary sequences of arbitrary operations (i.e. methods) can be defined in the action part of an ECA-rule, ARDBSs allow only arbitrary sequences of retrieval and modification operations (inclusive rollbacks).

2.3 Rule Management

Usually, a large number of rules exist in a database. In order to be efficient, an ADBS must organize these rules in an appropriate way, for example in collections of enabled and disabled rules. Thus, the number of rules that must be monitored at any given time can be controlled by activating and deactivating rules.

[2] Here, we use the REACH notation (cf. [Deu94]).

In ARDBS, rules are defined as metadata in the database schema and thus refer to a particular table. Similar to other metadata, operations are provided to add, drop, or modify rules. In contrast, rules in AOODBS are first class objects [DBM88]. Thus, rules can be organized in rule hierarchies, can have attributes (e.g. condition, and action), and can be created, deleted, modified, included into collections, related to other objects etc.

2.4 Rule Execution

The rule execution consists of two phases. In the first phase, which is triggered by the occurrence of the corresponding event, the condition of the rule is evaluated. If the evaluation yields true, the second phase which is the execution of the action part of the rule is started.

Both, condition evaluation as well as action execution are performed in transaction boundaries. These transactions are called *triggered transactions* whereas the transaction in which the triggering event occurs is called *triggering transaction.* The relationships between triggering and triggered transactions are defined in the rule execution model. One of these relationships is the commit and abort dependency, e.g. the triggering transaction may only commit if the triggered transaction commits. In addition, there are temporal dependencies concerning the triggering time of the triggered transaction, e.g. the condition must be evaluated immediately after the event occurrence.

Coupling modes, as introduced in HiPAC [DBB+88, CBB+89], determine how the rule parts are connected together. The *EC-coupling mode (or event-condition coupling mode)* determines when and in which transaction environment the condition is evaluated with respect to the triggering transaction. Analogous, the *CA-coupling mode (or condition-action coupling mode)* determines when and in which transaction environment the action is executed with respect to the triggering transaction. Up to now, the following coupling modes are proposed: *immediate*, *deferred*, *detached independent*, *detached dependent*, *detached sequential*, and *detached exclusive* (for details see e.g. [BBKZ94]). In the immediate case, the rule execution is started immediately (in a subtransaction). In the deferred mode, the rule execution is delayed until the end of the transaction. In the detached modes, the rule execution is performed in separate (detached or decoupled) transactions.

The deferred mode is useful, especially, for consistency maintenance where at the end of a transaction the consistency of the database has to be checked. The immediate mode, on the other hand, can be used for authorization control where the access rights of the user (application) must be checked before performing the requested operation.

In case several ECA-rules are defined on the same event, the ECA-rule execution component has to decide in which order these ECA-rules have to be triggered. For that, conflict strategies like priority-based ECA-rule ordering are used (cf. [DHW95, BZBW95]).

2.5 Architectures of Active Database Systems

In this section we recall the two main approaches for building active database systems: the *layered (or on-top)* approach and the *integrated (or built-in)* approach.

In the layered approach, all active components are built on top of a closed, passive database system. Due to the fact that the underlying database system is closed, the active components use the public interfaces as usual application programs do. SAMOS [Gat94], for example, provides active functionality on top of the object-oriented database system ObjectStore [LLOW91].

In contrast, in the integrated model active capabilities are integrated into the kernel of a database system. This approach is used, for example, by the prototype systems Sentinel [CKTB95] and REACH [BZBW95] which integrate active mechanisms into the object-oriented database system OpenOODB [WBT92].

So far, we have presented the essential notions and issues of active database systems needed to understand how active mechanisms can be used in a federated database environment for integrity constraint enforcement. In the following chapter, we briefly introduce federated database systems and some recent approaches to integrity maintenance in multi-database systems.

Chapter 3

Federated Database Systems

In this chapter, we first give a brief overview of concepts and notions of federated database systems following the survey in [SL90]. Then, we sketch recent approaches to integrity maintenance which seem to be relevant for federated databases as well.

3.1 Characterizing Federated Database Systems

A *federated database system* (FDBS) is a system which consists of a collection of cooperating component database systems (CDBSs) which are to a certain degree *autonomous* and possibly *heterogeneous*. These CDBSs interoperate and cooperate through a flexible federation layer which is on top of the CDBSs. Each CDBS can participate in the federation while continuing its local operations at the same time.

Three orthogonal dimensions characterize FDBSs: distribution, heterogeneity, and autonomy. Obviously, data is distributed over the CDBSs participating in the federation. Heterogeneity arises w.r.t. to different aspects. On the one hand, there can be heterogeneous database systems. Here, heterogeneity may concern for instance data models (i.e., the structures, inherent and explicit constraints, and query languages) or concerning the transaction management primitives and techniques. Concurrency control can be realized in different ways by different CDBSs, there may exists different commit protocols as well as different recovery policies. On the other hand, heterogeneity may arise as semantic heterogeneity which means that there is no agreement between the CDBSs (i.e., their administrators and users) w.r.t. to the meaning or interpretation of shared data as well as w.r.t. to the intended use of this data. Obviously, it is very difficult to detect and to deal with semantic heterogeneity because of its inherent character.

Due to the fact that there are slightly different understandings of what a federated database system is, we have to clearly state our view which goes along with [SL90]. Federated database systems are a particular kind of multi-database systems. A multi-database system consists of several, possibly heterogeneous database systems which cooperate or interoperate in some way. Therefore, a multi-database system is neither a centralized database system nor a distributed database system. Multi-database systems can be split

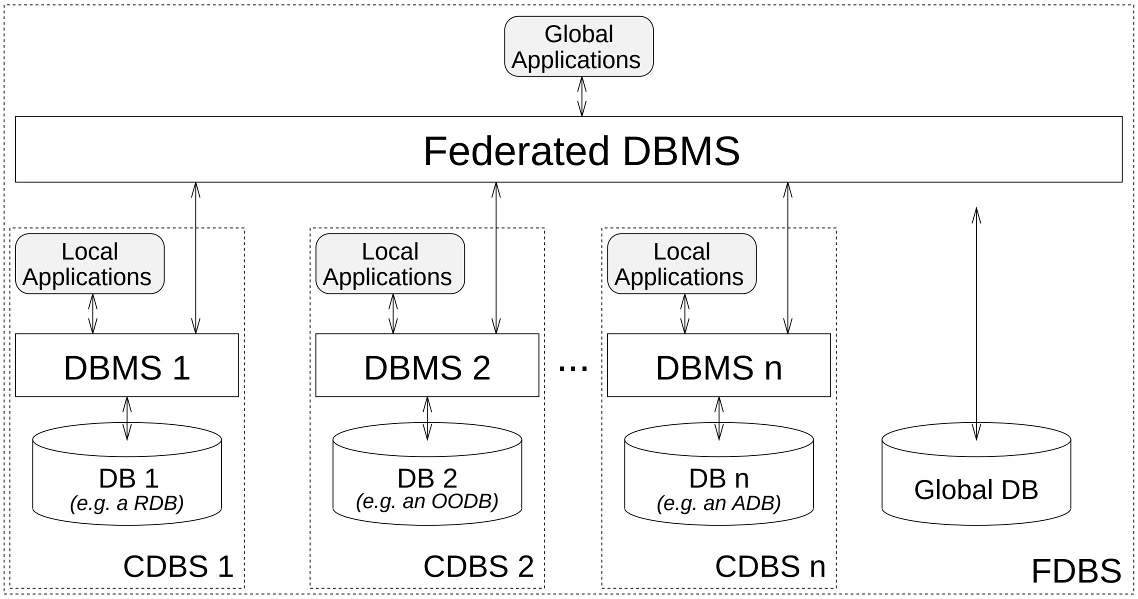

Figure 3.1: Architecture of federated database systems.

up into federated and nonfederated database systems. The latter ones are characterized by the property that their component DBSs are not autonomous. Hence, for a multi-database system to be a federated DBS we require a certain degree of autonomy for the component DBSs. In the following, we focus on tightly coupled FDBSs, because we are interested in global constraint maintenance in FDBS and it is quite obvious that the possibility of establishing a global constraint enforcement in loosely coupled FDBSs are rather limited. Moreover, it is usually not intended to establish global integrity constraints in loosely coupled FDBSs because there is no need for global constraints in typical application scenarios for loosely coupled FDBSs.

An architecture for tightly coupled FDBSs must provide an extensible federation layer (see Fig. 3.1). This layer provides a global interface for the *uniform* and *transparent* access to federated data. At the federation layer, there exists a global *common data model* (CDM) on which the schemata of the specific data models of the CDBSs can be mapped. Because of the requirements that a FDBS should support complex objects and has to be flexible (easy to extend), an object-oriented data model seems to be adequate as common data model. [RS94, BFN94], for instance, propose a model conforming to ODMG-93.

A central issue of FDBSs is that the independent CDBSs can continue performing their local operations while they are participating in a federation. In other words, CDBSs retain their interfaces so that existing applications can continue without any modifications.

In FDBSs two kinds of applications are distinguished: local and global (federated) applications. According to [SL90], global applications access federated data by sending requests to the FDBS while local applications submit their operations directly to a specific CDBS (cf. Fig. 3.1).

In the literature (e.g. [SL90, BHP92, LMR90]) several alternatives of architectures are discussed for FDBSs, e.g. with single or multiple federation. In this paper we consider a tightly coupled approach with a single federated schema. We choose this combination because it is best suited for maintaining global integrity of enterprise data. We are aware of the fact that any approach to global integrity restricts the autonomy of the CDBSs to a certain degree. But from an enterprise point of view, the integrity of federated data is a key issue, and therefore more important than the autonomy of local CDBSs of the departments.

3.2 Recent Work on Integrity Maintenance

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, e.g. [CGW94b, ÖV94]. In the sequel, we briefly survey some recent approaches towards global integrity enforcement in modern non-traditional database systems and point out how far they can be applied to federated database systems.

For object-oriented databases [JQ92] presents an approach to integrity maintenance based on a constraint compilation technique. This technique can be applied to declaratively specified global constraints and generates local constraints which can be maintained by object classes in an efficient way. The presentation in [JQ92] focuses on relational integrity, referential integrity and uniqueness requirements for objects. It seems that this approach can also be applied to multidatabase systems consisting of a number of homogeneous object-oriented database systems.

A special technique for distributed databases is described in [BG94]. *The Demarcation Protocol* allows to generate local constraints from a global constraint such that the satisfaction of the local constraints always implies the satisfaction of the original global constraint. The protocol enables a flexible treatment of the local constraints in a way that the local constraints may be modified in a controlled way. 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. As indicated in [BG94], the applicability of this protocol is mainly limited to arithmetic constraints.

The Reconfiguration Protocol [AE94] generalizes the Demarcation Protocol (as well as some other, similar protocols). By introducing so-called *reconfiguration transactions* which are intended to coordinate the modification of local constraints, the concurrency control mechanisms can be used for integrity maintenance.

In [GW93] another technique for local verification of global integrity constraints is presented. In case of local data modification it is possible to check locally whether the global constraint is violated. This approach can be applied to constraints which are expressed in a restricted first-order logic which ensures the practical evaluability of constraints. Although this technique seems to be a promising way of guaranteeing global integrity by local checks, it is not clear whether this approach can be applied to federated database systems where global transactions may cause modifications at the same time in different local databases.

A flexible way for constraint management in autonomous distributed databases is presented in [CGW94b, CGW94a]. Due to the autonomy of the involved database systems, it is necessary to provide a weaker notion of constraint enforcement. The proposed approach is based on interface specifications for the database systems. Such an interface specification consists of *guarantees* which the local database system promises to obey. A typical guarantee is for instance that whenever a certain write-request occurs, it is executed by the database system within a given time interval. Such guarantees can be expressed in a logical language providing adequate time-related constructs.

Similar to traditional database systems integrity maintenance in federated database systems has to work hand in hand with transaction management. Therefore, there is the indispensable prerequisite that the way of checking and enforcing integrity is adequate for the transaction model used in the database system. Several approaches to transaction management in multidatabase systems have been developed (for an overview see e.g. [BGS92]). A major topic for future research is the question which transaction model combined with which strategy of integrity enforcement is adequate for federated database systems.

Obviously, global integrity maintenance in federated database systems always requires a certain reduction of autonomy of the local database systems. Therefore, a way has to be found for introducing global integrity maintenance without restricting local autonomy too much. In the next chapter we present a pragmatic approach which seems to preserve a high degree of local autonomy by applying mechanisms of active databases on the global level for integrity maintenance and by making as much as possible use of existing local mechanisms for integrity maintenance.

Chapter 4

Integrating Active Mechanisms into a Federated Database System

According to the federated database system architecture presented in Chapter 3, we aim at providing a global mechanism to guarantee the integrity of federated data. In this chapter 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 CDBSs.

Our goal is to use ECA-rules in order to specify events, which potentially violate the integrity of the database, and (re)actions on these events, which should restore a consistent database state by aborting the operation/transaction (which is causing the event) or starting compensating (trans)actions.

Considering the costs for processing ECA-rules, it is not wise to realize every integrity constraint by an ECA-rule. For that, the existing local mechanisms for integrity maintenance should be used as much as possible in order to obtain an efficient solution. Only those integrity constraints which cannot appropriately be represented by the local integrity mechanisms should be realized by ECA-rules.

For that, we consider ECA-rules in the context of federated database systems in Section 4.1. Thereafter, we present a possible framework for an active FDBS in Section 4.2. Finally, we conclude this chapter by considering examples of active integrity enforcement in the proposed framework.

4.1 ECA-Rules in a Federated Database Environment

As we have seen in Chapter 2, the ECA-rule concept provides an uniform and powerful mechanism which can be used for integrity constraint enforcement, derived data maintenance, version control, and several other database features. In this section, we want to investigate how far ECA-rules can be adapted to a federated database environment. Here, we especially discuss how the specific rule parts have to look like in this case.

- *Necessary event types for (active) federated database systems.*

  According to [CM93, GD94b], an active system must support the specification and detection of various event types. Starting out from a common object-oriented data model at the federation level, only two primitive event types, method call and (absolute) time events, have to be distinguished (cf. [BZBW95]). This is due to the fact that in an object-oriented database system all operations, e.g. database and transaction operations, are realized in terms of methods.

  In addition to primitive events, an appropriate event algebra [CM93, GD94a, BZBW95] is needed for composing events by applying operators on primitive events. Here, the question is whether the proposed operators (see Section 2.2) are usable in federated database environments. Obviously, the conjunction and disjunction operators can be applied in the same way as usual. But, for the other proposed operators (sequence, closure, and history) there are semantic problems which are due to possible communication delays.

  Please note that in a federated database environment events may occur at two levels, i.e. in the CDBs or in the global database (GDB). Therefore, we need active components, e.g. event detectors, in the CDBSs as well as in the federation layer.

- *Possible conditions.*

  In federated database environments we need complex conditions which may refer to states of various CDBSs, for instance

  if (*object* **in** *DB1*) **and** (*object* **not in** *DB2*).

  Such complex conditions have to be subdivided into several (sub-) queries for the related CDBSs and the results of these queries must be merged at the federation level. In order to formulate complex conditions across several CDBSs, an algebra is required in which operators (e.g. and, or, not) for combining the results of subqueries are defined.

- *Admissible actions.* Here, user-defined methods as well as system-provided operations (like insert, update or delete) should be usable in the action part of an ECA-rule. Furthermore, predefined transactions or applications of a specific CDBS should be used as valid actions. It must also be possible to start global actions by calling methods or operations of different CDBSs. To realize such global actions, a FDBS must provide appropriate mechanisms for synchronizing actions.

- *Coupling modes.* The coupling modes of an ECA-rule determine when and in which transaction boundaries the condition and action parts will be executed. Therefore, the power and flexibility of an active system heavily depends on the coupling modes supported by the system. According to [BBKZ94], active systems have to support various coupling modes, especially in case of open environments. From our point of view, all proposed coupling modes are useful and needed for integrity enforcement purposes in federated database environments. In order to support these coupling modes, a FDBS must provide transaction mechanisms at the federation level. These mechanisms should contain at least open nested transactions to execute parts of global actions in subtransactions on CDBSs. Furthermore, in order to realize detached coupling modes transactions are needed which are able to create new top-level transactions.

4.2 Active Federated Database Framework

In this section, we propose an active federated database framework to provide a powerful and flexible platform for global integrity constraint enforcement in (tightly coupled) federated database systems.

Our main goal is to use active mechanisms in order to support global integrity constraints. Generally, global integrity constraints and (compensating) actions in case of violation can be formulated as ECA-rules in the following way:

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

Please notice that in an active federated database framework, an ECA-rule may also contain events and operations which refer to different CDBSs, e.g. a composite event which is a sequence of two primitive events e1 in the CDBS DB1 and e2 in the CDBS DB2.

As we have seen in Chapter 3, federated database systems 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 4.1). 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 (cf. [BZBW95]). These active components are integrated into the federation layer similar to the REACH approach.

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

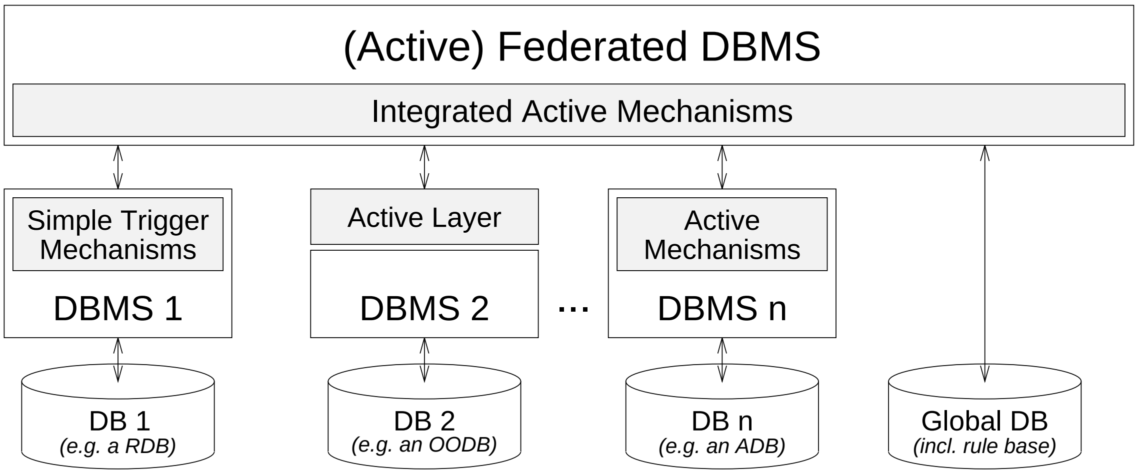

Figure 4.1: Active federated database framework

Therefore, we introduce the notion of *local ECA-rules* and *local rule managers*, respectively. 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 defined 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*

A general question is how can we adequately represent local ECA-rules as well as their local rule managers inside a FDBS? Obviously, for that we need some active mechanisms 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, for instance SYBASE [MD92] or ORACLE [Stü95], 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 as follows:

create trigger *LocalRuleForUpdateXXX*
update of <*attribute-name*>
on <*table-name*>
when <*predicate*>
<*operations-to-propagate-event*>

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 above.

2. *Providing an additional active layer with active mechanisms.* In case a CDBS does not support any active mechanisms, we have to provide an active layer á la SAMOS for that CDBS (see DBMS2 in Figure 4.1). 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 [Gat94, BZBW95]). However, we have to notice that method wrapping implies the recompilation of existing, *not with active capabilities equipped* application programs.

   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, respectively, the passing the operation to the original (passive) system.

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, for example

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

Such a global ECA-rule has to be subdivided into several local ECA-rules, i.e. in that case two local ECA-rules have to be generated.

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 subdivision of a composite event in several primitive events, and can be executed completely automatically.

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.

Applying active mechanisms for integrity enforcement in federated database environments seems to be a very promising way. In the following section, we present some examples to underline this statement.

4.3 Examples for Active Integrity Maintenance in a FDBS

In this section, we exemplarily show how active integrity maintenance in an active federated database environment can work. Especially, we present examples 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. In general, the resulting FBDS has multiple heterogeneous CDBSs as depicted in Figure 4.2. Let CDBS1 be a RDBS which supports simple trigger mechanisms, CDBS2 an OODBS with no active capabilities, and CDBSn an AOODBS. Further, let us assume that there exist several heterogeneous databases:

- CDBS1: is a relational database DB1 which contains several relations (i.e. tables), e.g. a relation Employee where all data of the employees of the enterprise are recorded;
- CDBS2: is an object-oriented database DB2 which consists of several classes, e.g. a class Project in which all data concerning the projects of that department are recorded;
- CDBSn: is an object-oriented database DBn which belongs to another department and contains similar classes as DB2.

Further, let us assume that all these local database schemas are integrated into a single federated schema, on which we can define global integrity constraints.

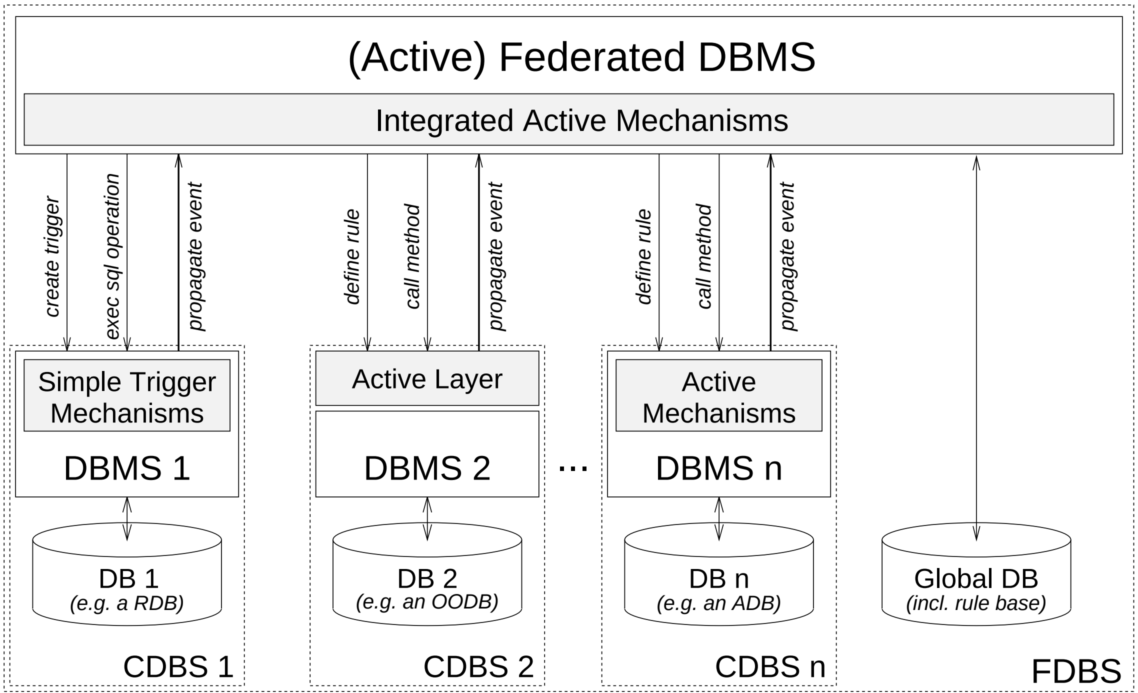

Figure 4.2: Rule definition and processing in an active FDBS

We now investigate how integrity constraints can be represented in the presented environment. Here, we have chosen a set of different kinds of integrity constraints to elucidate how active integrity maintenance can work in a federated database environment (see also Figure 4.2).

1. *Attribute constraints.* Attribute constraints restrict the values an attribute can have. Such constraints are represented, for instance, in relational database systems by the well-known **check** or **not null** clauses.

   In our sample scenario, we may have a 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 federated CDBSs. Otherwise, we may realize it by a global 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, 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 a FDBS has to be identifiable not only in the CDBS but also in the whole FDBS. In order to guarantee the global uniqueness of a FDBS object, global key constraints may be defined, e.g. each project in the enterprise is identified by 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 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 *global action(s) which restore(s) a consistent FDB state*

3. *Aggregate constraints.* Such constraints concern generally several objects, e.g. the sum of the budgets of all projects in the enterprise must not exceed a certain amount. Aggregate constraints can be realized similar to the key constraints:

   define rule *GlobalAggregateConstraintRule*

on *budget manipulation operations*
if *sum of budgets exceed a given amount*
do *reject or compensate the corresponding operation*

4. *Referential integrity constraints.* Referential integrity constraints specify that a database object refers to an existing object. In our sample scenario, we may have the constraint that each participant of a project must be an employee of the enterprise at the same time. The following ECA-rule, for example, guarantees that only employees can be involved in the projects of the enterprise.

   define rule *GlobalReferentialIntegrityRule*
   on *deletion of an employee*
   if *this employee is involved in a project*
   do *remove this participant from the project*

5. Further constraints: 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.

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

Chapter 5

Conclusions

Our proposal for integrity enforcement in a federated, heterogeneous framework is based on the idea to bring active mechanisms into (tightly coupled) federated database systems. The active rule paradigm known from active databases is a powerful way for specifying integrity constraints and enforcing them. We have shown in which way active rules can be applied in FDBSs for global constraint maintenance. Our approach respects the local autonomy of the component database systems as much as possible, especially by using integrity mechanisms offered by them. Obviously, preserving local autonomy and enforcing global constraints represent two contrary requirements. Therefore, global integrity constraints should be used only in a well thought out and restricted way and the corresponding global ECA-rules should be designed very carefully.

As foundation for global integrity enforcement, a FDBS must support an appropriate transaction model which allows the co-existence of local and global (federated) transactions. Another important issue is that the architecture of the FDBS has to be open so that it is possible to add/remove CDBSs into/from a FDBS or to include new transaction managers which realize more sophisticated transaction paradigms.

Beside the development of a prototype system for validating our approach in practice, there are several other questions and problems we are going to work on. One problem is the derivation of local ECA-rules from global ECA-rules. A global ECA-rule referring to a composite event has to be subdivided in several local ECA-rules, for instance one local ECA-rules for each local event. 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. Obviously, local events can be composite events themselves. Therefore, the derivation of local ECA-rules from global ones must be done very carefully. In addition, there is an interesting question whether parts of the global condition (in the global ECA-rule) can be delegated to local ECA-rules. Thereby, we could take some of the load off the federation layer. Nevertheless, there are semantical problems because the local conditions are evaluated at another instant of time.

Another very important question is related to the design of global integrity constraints. The definition of global integrity constraints has to be done very carefully. The rule designer must be informed about the existing local integrity constraints in order to avoid unnecessarily redundantly defined constraints, and thus avoiding multiple checks of one and the same constraint. Another important aspect is that the rule designer has to consider the collaboration of global and local integrity constraints in order to avoid contradictory actions.

These are only two of several important problems we are investigating. For this we have just started a project in which we will put our ideas of global integrity maintenance for federated database systems into practice. This interdisciplinary project deals with the problem of cooperative, distributed planning of factories. Currently, a large number of software tools like CAD systems is used in such a planning process. These tools come along with different database systems (or, even worse, with their own data management systems) or offer interfaces to different database systems. Due to this heterogeneity w.r.t. data management, a federation of the database systems being involved is needed for improving efficiency and reliability of the planning process. Such a federated database system could be an ideal basis for implementing a workflow system for factory planning. Global constraint management and integrity maintenance are absolutely necessary for this application area in order to adequately manage the data which is produced by several designers (engineers) at the same time and which is highly interrelated. For a prototype implementation of our active federated database system framework we are cooperating with Cadlab Paderborn (a joint venture of University of Paderborn and Siemens-Nixdorf). Cadlab is currently developing a middleware software called OpenDM (partly described in [RS94, RS95, RBB+95]) which provides all necessary functionalities for federating heterogeneous database systems.s

Bibliography

[AE94] 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.

[BBKZ92] A. P. Buchmann, H. Branding, T. Kudrass, and J. Zimmermann. REACH: A REal-Time, ACtive and Heterogenous Mediator System. *Bulletin of the IEEE Technical Committee on Data Engineering*, 15(1-4):44–47, December 1992.

[BBKZ94] H. Branding, A. Buchmann, T. Kudrass, and J. Zimmermann. Rules in an Open System: The REACH Rule 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 111–126. Workshops in Computing, Springer-Verlag, 1994.

[BFN94] R. Busse, P. Fankhauser, and E. J. Neuhold. Federated Schemata in ODMG. In J. Eder and L. A. Kalinichenko, editors, *Extending Information Systems Technology – Proc. of the 2nd Int. East/West Database Workshop, Klagenfurt, Austria*, pages 356–379. Workshops in Computing, Springer-Verlag, September 1994.

[BG94] 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, 1994.

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

[BHP92] M. Bright, A. Hurson, and S. Pakzad. A Taxonomy and Current Issues in Multidatabase Systems. *IEEE Computer*, 25(3):50–56, March 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. Springer-Verlag, 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.

[CBB+89] S. Chakravathy, B. Blaustein, A. Buchmann, M. Carey, U. Dayal, D. Goldhirsch, M. Hsu, R. Jauhari, R. Ladin, M. Livny, D. McCarthy, R. McKee, and A. Rosenthal. HiPAC: A Research Project in Active, Time-Contrained Database Management. Technical Report XAIT-89-02, Xerox Advanced Information Technology, 1989.

[CGW94a] S. Chawathe, H. Garcia-Molina, and J. Widom. Constraint Management in Loosely Coupled Distributed Databases. Technical Report, Department of Computer Science, Stanford University, 1994.

[CGW94b] 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.

[Cha89] S. Chakravarthy. Rule Management and Evaluation: An Active DBMS Perspective. In J. Clifford, B. Lindsay, and D. Mayer, editors, *Proc. of the 1989 ACM SIGMOD Int. Conf. on Management on Data, Portland, Oregon, SIGMOD RECORD 18(2)*, pages 20–28. ACM Press, June 1989.

[CKAK94] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite Events for Active Databases: Semantics, Contexts and Detection. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB'94), Santiago, Chile*, pages 606–617. Morgan Kaufmann Publishers, 1994.

[CKTB95] S. Chakravarthy, V. Krishnaprasad, Z. Tamizuddin, and R. H. Badani. ECA Rule Integration into an OODBMS: Architecture and Implementation. 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 341–348. IEEE Computer Society Press, March 1995.

[CM93] S. Chakravarthy and D. Mishra. Snoop: An Expressive Event Specification Language for Active Databases. Technical Report UF-CIS-TR-93-007, Computer Science Department, University of Florida, March 1993.

[DBB+88] U. Dayal, B. Blaustein, A. Buchmann, S. Chakravarthy, D. Goldhirsch, M. Hsu, R. Ladin, D. McCarthy, and A. Rosenthal. The HiPAC Project: Combining Active Databases and Timing Contraints. *ACM SIGMOD RECORD*, 17(1):51–70, March 1988.

[DBM88] U. Dayal, A. Buchmann, and D. McCarthy. Rules are Objects too: A Knowledge Model for an Active Object-Oriented Database System. In K. R. Dittrich, editor, *Proc. of the 2nd Int. Workshop on Object-Oriented Database Systems (OODBS'88), Bad Münster, Germany*, pages 129–143. LNCS 334, Springer-Verlag, September 1988.

[Deu94] A. Deutsch. Method and Composite Event Detection in the "REACH" Active Database System. Master's thesis, Technical University of Darmstadt, 1994.

[DG93] K. R. Dittrich and S. Gatziu. Time-Issues in Active Database System. In R. T. Snodgrass, editor, *Proc. of the Int. Workshop on an Infrastructure for Temporal Databases, Arlington, Texas, USA*, June 1993.

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

[Gat94] S. Gatziu. *Events in an Active, Object-Oriented Database System.* Ph.D. thesis, Verlag Dr. Kovač, Hamburg, 1994.

[GD92] S. Gatziu and K. R. Dittrich. SAMOS: an Active Object-Oriented Database System. *Bulletin of the IEEE Technical Committee on Data Engineering*, 15(1–4):23–26, December 1992.

[GD94a] S. Gatziu and K. R. Dittrich. Detecting Composite Events in Active Database Systems Using Petri Nets. In J. Widom and S. Chakravarthy, editors, *Proc. of the 4th Int. Workshop on Research Issues in Data Engineering: Active Database Systems (RIDE-ADS'94), Houston, Texas, USA*, pages 2–9. IEEE Computer Society Press, February 1994.

[GD94b] 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.

[GJ91] N. H. Gehani and H. V. Jagadish. Ode as an Active Database: Constraints and Triggers. In G. M. Lohmann, A. Sernadas, and R. Camps, editors, *Proc. of 17th Int. Conf. on Very Large Data Bases (VLDB'91), Barcelona, Spain*, pages 327–336. Morgan Kaufmann Publishers, September 1991.

[GJS92a] N. Gehani, H. V. Jagadish, and O. Shmueli. Event Specification in an Active Object-Oriented Database. In M. Stonebraker, editor, *Proc. of the 1992 ACM*

SIGMOD Int. Conf. on Management of Data, San Diego, CA, SIGMOD RECORD 21(2), pages 81–90. ACM Press, June 1992.

[GJS92b] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Composite Event Specification in Active Databases. In L.-Y. Yuan, editor, *Proc. of the 18th Int. Conf. on Very Large Data Bases (VLDB'92), Vancouver, Canada*, pages 265–276. Morgan Kaufmann Publishers, August 1992.

[GW93] 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 on Data, Washington, D.C., SIGMOD RECORD 22(2)*, pages 49–58. ACM Press, June 1993.

[Han89] E. N. Hanson. The Initial Report on The Design of Ariel: A DBMS With an Integrated Production Rule System. *ACM SIGMOD RECORD*, 18(3):12–19, September 1989.

[Han92] E. N. Hanson. Rule Condition Testing and Action Execution in Ariel. In M. Stonebraker, editor, *Proc. of the 1992 ACM SIGMOD Int. Conf. on Management of Data, San Diego, CA, SIGMOD RECORD 21(2)*, pages 49–58. ACM Press, June 1992.

[JQ92] H. V. Jagadish and X. Qian. Integrity Maintenance in an OODB. In L.-Y. Yuan, editor, *Proc. of the 18th Int. Conf. on Very Large Data Bases (VLDB'92), Vancouver, Canada*, pages 469–480. Morgan Kaufmann Publishers, August 1992.

[LLOW91] C. Lamb, G. Landis, J. Orenstein, and D. Winreb. The ObjectStore Database System. *Communications of the ACM*, 34(10):50–63, October 1991.

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

[MD92] D. McGoveran and C. J. Date. *A Guide to Sybase and SQL Server*. Addison-Wesley, Reading, MA, 1992.

[ÖV94] M. T. Özsu and P. Valduriez. Distributed Data Management: Unsolved Problems and New Issues. In T. Casavant and M. Singhal, editors, *Readings in Distributed Computing Systems*, chapter 10, pages 512–544. IEEE Computer Society Press, 1994.

[PGW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. 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 261–260. IEEE Computer Society Press, March 1995.

[RBB+95] E. Radeke, R. Böttger, B. Burkert, Y. Engel, G. Kachel, S. Kolmschlag, and D. Nolte. Efendi: Federated Database System of Cadlab. In M. J. Carey and D. A. Schneider, editors, *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, SIGMOD RECORD 24(2)*, page 481. ACM Press, June 1995.

[RS94] E. Radeke and M. H. Scholl. Federation and Stepwise Reduction of Database Systems. In W. Litwin and T. Risch, editors, *Proc. of the 1st Int. Conf. on Applications of Databases (ADB'94), Vadstena, Sweden*, pages 381–399. LNCS 819, Springer-Verlag, June 1994.

[RS95] E. Radeke and M. H. Scholl. Functionality for Object Migration among Distributed, Heterogenous, Autonomous Databases Systems. In O. Bukhres, T. Özsu, and M.-C. Shan, editors, *Proc. of the 5th Int. Workshop on Research Issues in Data Engineering: Distributed Object Management (RIDE-DOM'95), Taipei, Taiwan*, pages 58–66. IEEE Computer Society Press, March 1995.

[SHH87] M. Stonebraker, E. Hanson, and C.-H. Hong. The Design of the POSTGRES Rules System. In B. Wah, editor, *Proc. of the IEEE Int. Conf. on Data Engineering (ICDE'87)*, pages 365–374. IEEE Computer Society Press, February 1987.

[SHP88] M. Stonebraker, E. Hanson, and S. Potamianos. The POSTGRES Rule Manager. *IEEE Transaction on Software Engineering*, 14(7):897–907, July 1988.

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

[Stü95] G. Stürner. *ORACLE7 – A User's and Developer's Guide.* International Thomson Publishing, London, 1995.

[TC95] C. Türker and S. Conrad. Aktive Mechanismen zur Konsistenzsicherung in föderierten Datenbanksystemen. In C. Eckert, H.-J. Klein, and T. Polle, editors, *7. GI-Workshop "Grundlagen von Datenbanken", Bad Salzdetfurth*, pages 148–152. Hildesheimer Informatik Berichte, June 1995.

[WBT92] D. L. Wells, J. A. Blakeley, and C. Thompson. Architecture of an Open Object-Oriented Database Management System. *IEEE Computer*, 25(10):74–82, 1992.

[WCL91] J. Widom, R. J. Cochrane, and B. G. Lindsay. Implementing Set-Oriental Production Rules as an Extension to Starburst. In G. M. Lohmann, A. Sernadas, and R. Camps, editors, *Proc. of the 17th Int. Conf. on Very Large*

Data Bases (VLDB'91), Barcelona, Spain, pages 275–285. Morgan Kaufmann Publishers, September 1991.

[WF90] J. Widom and S.J. Finkelstein. Set-Oriented Production Rules in Relational Database Systems. In H. Garcia-Molina and H. Jagadish, editors, *Proc. of the 1990 ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, SIGMOD RECORD 19(2)*, pages 259–270. ACM Press, June 1990.