Extending the Schema Architecture of Federated Database Systems for Replicating Information in Hospital Information Systems

W. Hasselbring
University of Dortmund, Department of Computer Science, Informatik 10
D-44221 Dortmund, Germany
E-mail: `willi@ls10.informatik.uni-dortmund.de`

Abstract

Some problems and solutions for propagation of information updates across heterogeneous subsystems within hospitals are discussed. The presented extended schema architecture for federated database systems is the basis for algorithms that restore the integrity of replicated information when changes occur.

1 Introduction

The purpose of a hospital information system (HIS) is to manage the information that health professionals need to perform their jobs effectively and efficiently [SPFW90]. Integrated systems which satisfy all requirements on information processing in hospitals are not available; even if some vendors promise this. Also, from an economical perspective, it is desirable to install a number of applications, which effectively support the specific needs of the individual organizational units of a hospital. Typical examples are systems for patient registration, admission, discharge and transfer, appointment scheduling, management of laboratory tests as well as decision support for medical treatment. This situation naturally leads to a collection of heterogeneous subsystems scattered across the hospital. To effectively support the work in hospitals, it is necessary to integrate these subsystems avoiding multiple entry of the same information and inconsistencies among information that is stored in different subsystems. Integration is a decisive factor for the successful operation of a computer-based HIS [ESP92]. The integration of data from various sources in the hospital produces a rich database supporting health professionals with their work. A modular system of interoperable and cooperating subsystems, which retain their autonomy as far as reasonable, is required.

As a small portion, Figure 1 illustrates the overlapping areas of information relevant for individual subsystems in hospitals for a laboratory, a radiology and an administration subsystem. It is important to note that it is not the goal to provide access from all places in the hospitals to all the information that is relevant for the hospital, but to integrate the overlapping areas. The basic patient data such as name and birthday are in the overlapping area of all systems, but insurance information and therapy results are only relevant to some subsystems. Note, however, that the sizes of the areas in Figure 1 are not proportional; only the structural segmentation is illustrated.

A major need of HISs is, therefore, the integration of the overlapping areas of information stored among different and heterogeneous applications, even if they have been developed at different times, by different vendors and with different technologies. An open federation of autonomous but interworking systems should provide optimized support to the specific needs of the individual units by enabling different vendors to offer specialized applications and allowing the users to select the most effective solutions for their needs.

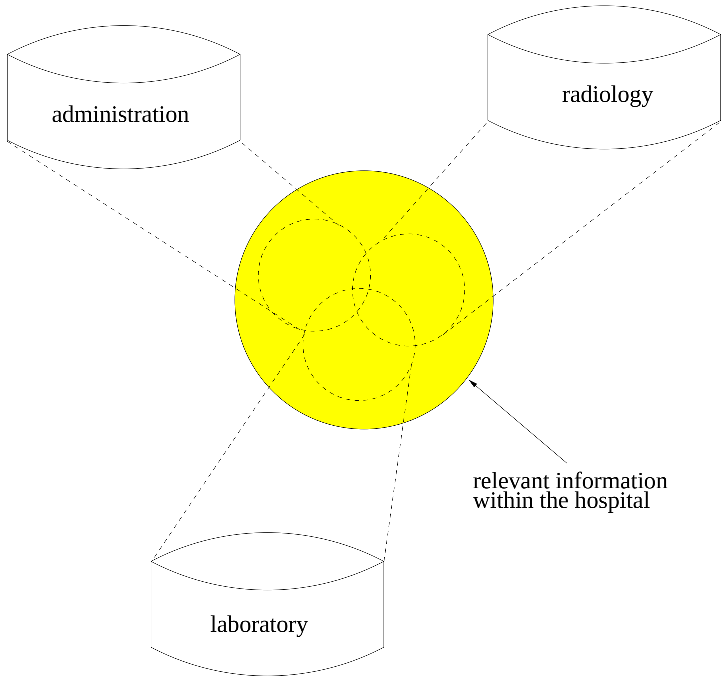

Figure 1: The overlapping areas of information stored among subsystems in hospitals.

However, most current work on federated database systems is concerned with offering access for global applications to information that is stored in several dissimilar systems. Another concern is integrity maintenance across autonomous systems [CT95]. In hospitals, replicating information among several autonomous subsystems is a central issue.

The purpose of this paper is to study some problems and solutions of propagation of information updates across heterogeneous subsystems within hospitals. The general structure of our presented architecture is based on the reference architecture for federated database systems [SL90] and adapted to the specific demands on integration of replicated information. This architecture is the basis for algorithms that restore the integrity of replicated information when changes occur.

The current state of the art in connecting subsystems within hospitals through communication servers is discussed in Section 2 and our software architecture for integration of replicated information within hospitals is presented in Section 3. Section 4 discusses some related work and Section 5 draws some conclusions.

2 Current state of the art: connecting subsystems within hospitals through communication servers

To connect heterogeneous subsystems in hospitals, communication servers are often deployed [PD95]. Figure 2 displays an example configuration of a HIS with a central communication server. In this configuration, a laboratory, a radiology, two wards, an administration, and a pharmacy application are connected by the server. The communication server enables the subsystems to send message to each other. Each subsystem is connected to the communication server and sends messages only to this server. The communication server determines the receiver and forwards the message. Hospital communication servers usually support standard protocols such as HL7 [Ham93] and the translation across different protocols when forwarding messages.

The requirement for building complex systems that combine heterogeneous subsystems can be met at the low level of *interconnectivity* or at the higher level of *interoperability* [PBE95].

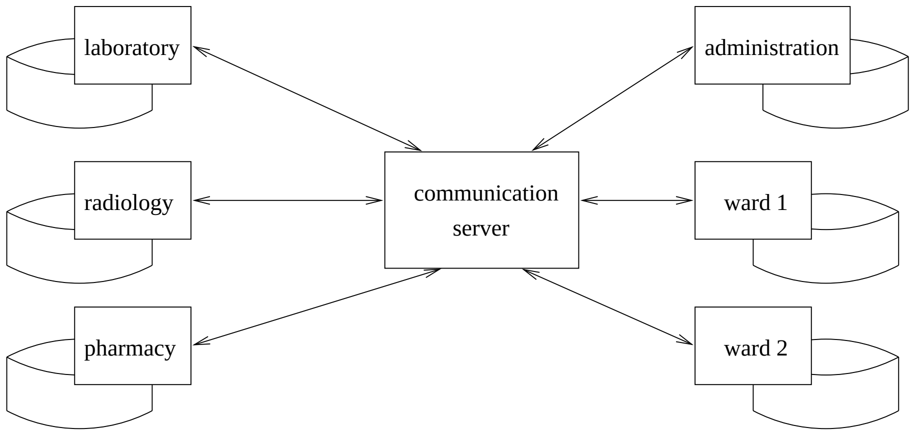

Figure 2: A possible configuration for the integration of a distributed HIS through a communication server.

Interconnectivity simply supports system *communication*, while interoperability additionally supports systems to cooperate in the joint execution of tasks. A communication server only supports interconnectivity: the subsystems themselves

- need to know where to send which messages,
- must take the initiative to update replicas and send messages for this purpose, and
- must be aware to receive messages from other systems and store the message contents appropriately in their local data stores.

With an integration that is based on a communication server, it is *not* known at the integration level at which sites data actually is *stored*. It is only known that data is *exchanged*. A communication server does not know whether data is replicated or just needed temporarily by a client for answering a user query.

With a tightly coupled federated database system whose data integration is on the basis of schema integration, the federation layer is capable of supporting subsystems to interoperate. Instead of enabling the subsystems with a communication server to send messages for information exchange, the exchange of information can be accomplished through updates of replicated data in subsystems by the federation layer as will be discussed in Section 3.

3 A federated software architecture for integrity maintenance of replicated information

A *database system* (DBS) consists of a database management system and one or more databases that it manages. In a federated DBS, both global applications and local applications are supported [SL90]. The local applications remain autonomous, but must restrict their autonomy to some extent to participate in the federation. Global applications can access multiple local DBSs through the federation layer. The federation layer can also control global integrity constraints such as data value dependencies across multiple component DBSs.

This section presents our federated software architecture which has been designed according to the specific requirements of integrating replicated information among heterogeneous HISs. The following subsections present an extended schema architecture and the associated algorithms that restore the integrity of replicated information when changes occur.

3.1 The schema architecture

For federated DBSs, the traditional three-level schema architecture [Dat95] must be extended to support the dimensions of distribution, heterogeneity, and autonomy. The generally accepted reference architecture for schemas in tightly coupled federated DBSs is presented in [SL90] and, in the same form, in [PBE95] where approaches to object-orientation in multidatabase systems are surveyed. The diagram in Figure 3 displays this schema architecture which presents, apart from the dots that indicate repetition, one possible configuration of a federated database system. The edges between the schemas in Figure 3 correspond to software processors as indicated in the right hand column of Figure 3.

As discussed in [SL90], several options in the schema and processing architecture are available, some of which are:

- Any number of external schemas can be defined, each with its own filtering processor.
- Any number of federated schemas can be defined, each with its own constructing processor.

  A tightly coupled federated DBS with multiple federations allows the tailoring of the use of the federated DBS with respect to multiple classes of federation users with different data access requirements.
- Schemas on all levels, except the local and federated schemas, are optional and may be combined into a single schema of another level.
- A component DBS can participate in more than one federation and continue the operation of local applications.

Note, that a schema architecture which consists of just one federated schema and some local schemas concurs with the 5-level schema architecture of [SL90]. The other levels contain no schemas in this case.

These constraints are not defined formally in [SL90]. Below, an extended model for our tightly coupled federated DBS architecture will be presented in a semi-formal way by means of an object-oriented modeling technique.

It is rather obvious that the reference schema architecture has been designed primarily to support global access to the component DBSs, only secondarily to support integrity control. Therefore, we extend the reference schema architecture of [SL90] with import, export and import/export distinction for *public* schemas to adequately support the algorithms for changing replicated information. Figure 4 displays a *generic* metamodel for this extended schema architecture for federated DBSs using the Unified Modeling Language (UML) notation for class diagrams [RSC97]. In this model, some of the constraints and options for the architecture are defined by means of the *cardinalities* at the associations. The distinct classes of *Public Schemas* replace the export schemas in the reference architecture of [SL90].

Specifying an import schema in our architecture is a subscription to change notifications for the corresponding data. Export schemas specify data to be exported to other systems. Import/export schemas define data to be both imported and exported. The schema types determine the change algorithms for integration of replicated information as will be discussed below.

To explain the diagram in Figure 4: Rectangles are the UML symbols for classes. In UML, cardinalities for associations are specified through numerical ranges at the association links. The default cardinality is 1. If the cardinality specification comprises a single star, then it denotes the unlimited non-negative integer range (zero or more). The arrows attached to the association names indicate the direction for reading the names which are annotations to associations (called *name direction*) [RSC97].

The association between local schema and component schema in Figure 4 specifies that each component schema is transformed from exactly one local schema, but each local schema

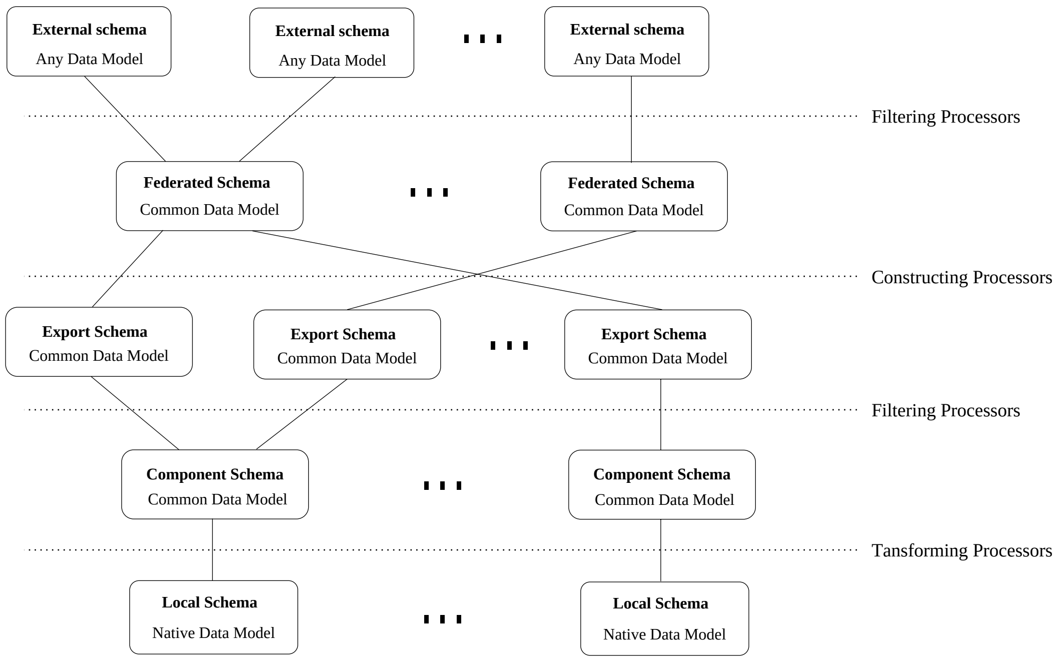

Figure 3: The 5-level schema architecture as presented in [SL90] and annotated with the corresponding processor types.

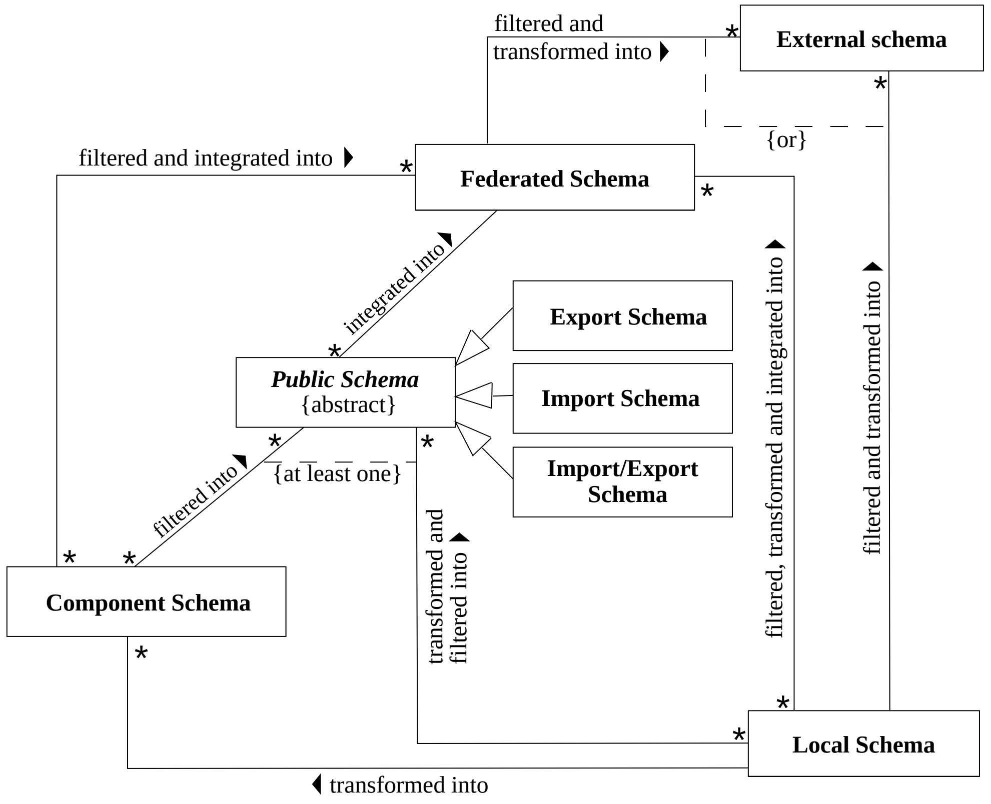

Figure 4: Modeling the extended 5-level schema architecture as a UML class diagram [RSC97].

can be transformed into multiple component schemas when the corresponding component DBS participates in more than one federation.

A constraint in UML is a semantic relationship among model elements that specifies conditions which must be maintained [RSC97]. A constraint represents semantic information attached to a model element syntactically enclosed in braces. The predefined or-constraints indicate situations in which only one of several potential associations may be instantiated at one time for any single object. This is shown as a dashed line connecting two or more associations, all of which must have a class in common, with the constraint {or} labeling the dashed line. Any instance of the class may only participate in at most one of the associations at one time. This is simply a particular use of the constraint notation.

Each *Public Schema* is filtered from at least one component or local schema and integrated into exactly one federated schema. Each external schema is derived from either one federated or one local schema, etc. External schemas which are directly derived from local schemas are used for local applications.

Inheritance is shown in UML as a solid-line path from the sub-class to the super-class, with a large hollow triangle at the end of the path where it meets the super-class [RSC97]. In Figure 4, *Public Schema* is an abstract class [Mey88]. The concrete classes Export Schema, Import Schema, and Import/Export Schema inherit all associations from *Public Schema*. There will be no instances (schemas) of the abstract class *Public Schema* in an instantiated schema architecture.

The diagram in Figure 3 (apart from the dots that indicate repetition) can be regarded as an instance of the model in Figure 4. The model in Figure 4 is a *meta model* for schemas and their associations.

3.2 Change algorithms

Our schema architecture is the basis for algorithms that restore the integrity of replicated information when changes occur. For generality, we use the term *change* for insertion, deletion and update of data. Below, a change algorithm with one master copy for data items and a change algorithm with multiple master copies for data items are motivated and discussed. In the sequel, the specification of change propagation and the detection of changes are discussed.

3.2.1 Change algorithm with one master copy for data items

As discussed in [SWG$^+$92], each datum in a distributed DBS for electronic medical records (and, consequently, in a HIS) should have only one *master copy* at which changes are allowed. Note, however, that such a master (server) can cooperate with multiple clients that intend to modify the datum. The restriction to one master copy does not imply a restriction for data entry from just one location within a hospital.

A system that is the client in one situation may be the server in another situation provided only one system is the master for a particular datum. A datum may be allowed to reside in multiple places (replica), but changes must be handled through the master who forwards the changes to all places where copies of this datum exist.

Figure 5 illustrates an example scenario for changing replicas. Component DBS 1 exports a datum through an export schema. Data about the same real world phenomenon is stored in component DBSs 2, 3 and 4. The latter three component DBSs import this datum through some import schemas. Component DBSs 2 and 3 share the same import schema. To integrate the semantic replication of the same real world phenomenon, the federated schema relates the corresponding parts of export schema 1 to the corresponding parts of import schemas 2 and 3. A change event in component DBS 1 on an exported data item triggers corresponding change operations of the replicas which are imported by the other three component DBSs. The dotted lines illustrate the data flow.

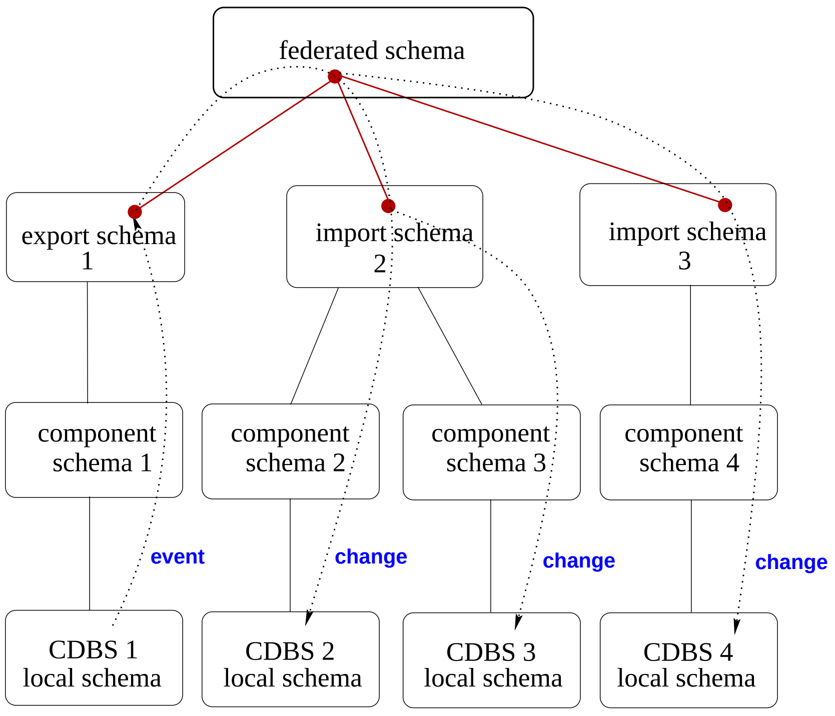

Figure 5: An example scenario for changing replicas. The model in Figure 4 is the *meta model* for the schemas and their associations in this scenario. The dotted lines illustrate the data flow.

The federated schema relates elements of export and import schemas to each other, in which each element relates exactly one export element to one or more import elements. This constraint should be enforced by the integration tools.

3.2.2 Change algorithm with multiple master copies for data items

There exists the need to integrate pre-existing legacy database and file systems into HISs. Typically, these legacy information systems have evolved over many years and play a crucial role in the day-to-day information processing of the hospital. They are often difficult to modify and virtually impossible to rewrite.

There is, therefore, a need to provide techniques not only for accessing the data which is locked inside these systems from newer systems, but also for providing a strategy which allows the gradual migration of the systems to new platforms and architectures. A *smooth* migration from legacy systems to modern information systems can be accomplished with federated DBSs [RS95].

To integrate replicated information across legacy systems, it cannot be expected that only one master copy for each datum exists at which changes are allowed, because legacy systems usually store the data in their own repositories where the data items must be considered as master copies. To incorporate such situations in which multiple master copies for specific data items are needed, the import/export schemas can be used in our architecture. An import/export schema specifies that the corresponding data items are imported as well as exported.

The difference to a combination of an import with an export schema is that data which is changed by the federation layer does not trigger additional changes to be propagated by the federation layer. Only changes by local applications trigger change events to be propagated by the federation layer.

This mechanism avoids endless loops of changes by the federation layer when the same information is exported as well as imported by multiple component DBSs. However, import/export

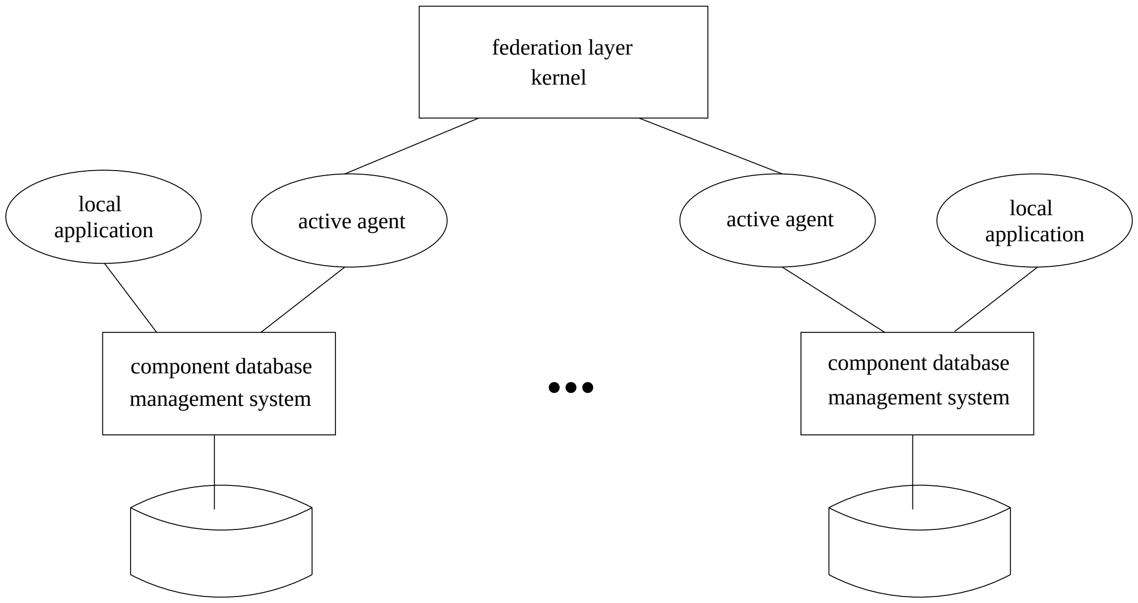

Figure 6: Active agents in our architecture. Global applications are not displayed in this figure, but they could be connected to the federation layer.

schemas should only be used when multiple master copies for specific data items are required. For import/export schemas we do not have the constraint that only one data source is allowed.

3.2.3 Specification of change propagation

For the specification of change mechanisms, *agents* connected to the component DBSs are introduced as *active* DBSs [WC96]. Figure 6 illustrates this division of labor between kernel and agents. The local database management systems of the component DBSs consider the active agents as local applications.

This way, the kernel of the federation layer sees the component DBSs as active DBSs. An active DBS is an extended conventional DBS which has the capability to monitor predefined situations (situations of interest) and to react with defined actions [WC96]. Such re-active behavior is generally expressed by the so-called *event-condition-action* rules (ECA rules) which define what to do if a certain situation occurs in the DBS. ECA rules are a promising principle not only for integrity enforcement in single, centralized DBSs, but also for federated DBSs [CT95]. The active rule mechanism can be considered as a communication mechanism between the component DBSs and the federation layer. Therefore, it is rather straightforward to use ECA rules to specify integrity constraints for replicas and actions which have to be executed in case of potential integrity violations.

Let *ExportSchemas* denote the set of export schemas, *ImportSchemas* denote the set of import schemas and *ImExSchemas* denote the set of import/export schemas. The change mechanisms for our architecture are specified as follows:

```
$\forall$ ES: $\mathit{ExportSchemas} \cup \mathit{ImExSchemas}$:
  on    $\mathit{event}$(ES)
  if    $\exists$ IS: $\mathit{ImportSchemas} \cup \mathit{ImExSchemas} \mid \mathit{dependence}$ (ES, IS)
  then  -- change dependent values:
        $\forall$ IS $\in \mathit{ImportSchemas} \cup \mathit{ImExSchemas} \mid \mathit{dependence}$ (ES, IS) : $\mathit{change}$ (IS)
```

Note, however, that this is only a superficial specification of the general mechanisms. For a detailed specification, it would be necessary to specify the structure of the schemas and the functions *event*, *dependence* and *change* which operate on the schemas. The *change* function must not raise events on *ImExSchemas*. A detailed and exhaustive formal specification is beyond the scope of the present paper which focuses on the overall system architecture and the general mechanisms of the associated algorithms.

The execution of rules 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 and action execution, are performed in transaction boundaries. These transactions are called *triggered transactions* whereas the transaction in which the event occurs is called *triggering transaction*. Coupling modes between triggering and triggered transactions determine when the triggered transactions are executed [WC96]. For our approach, the *decoupled* mode is most reasonable, as we should not restrict the autonomy of component DBSs more than necessary.

In HIS, there occur predominantly insertions of new information; modifications of existing information occur very seldom. Therefore, a weaker consistency criterion is acceptable: you rarely see outdated information that has been updated somewhere else. You only see new inserted information later on. Therefore, it is reasonable to execute the change operations in separate transactions in this environment. Furthermore, immediate and deferred coupling would restrict the autonomy substantially.

Rule processing is subject to infinite loops, that is, rules may trigger one another indefinitely. In general, it is an undecidable problem to determine in advance whether rules are guaranteed to terminate, although conservative algorithms have been proposed that warn the rule programmer when looping is possible [AHW95]. A prevention against infinite loops in our architecture is the prohibition of cycles in dependencies among import and export schemas via component and federated schemas.

3.2.4 Detecting changes by the active agents

How do the agents find out about changes to data? To solve this problem, a balance between autonomy and integration must be found. Some approaches are:

- Some DBSs offer active mechanisms such as *triggers* to detect and announce changes [WC96]. With the availability of active mechanisms, local applications do not have to be changed: triggers are assigned to monitor changes of exported data.
- If a component DBS does not support such detecting techniques, *polling* techniques can be deployed:
  - The evaluation of system data can be used to detect the specific operations. For instance, the transactions log file can be monitored [EK91].
  - Changes can be detected by comparing data snapshots. Keys can be used to efficiently compute the changes, as described in [LGM96].
- In client/server systems, an interface between application and server can be used to analyze the client requests and announce detected changes [KLB96].
- If the component DBS is an object-oriented DBS, the stored objects can be modified by an overriding technique [SS95]. Any critical method will have to be refined by adding operations that announce changes. This approach restricts the autonomy of local applications, since the local applications are changed.

However, a promising approach in a hospital setting is the following:

- Wrapping HL7-messages. HL7 is a de-facto standard for data exchange between commercial systems for hospitals [McD95].

A HL7 message is a string, which contains mandatory and optional segments [Ham93]. These segments consist of several fields. The syntax of version 2.2 of HL7 messages is defined informally in [HLG94]. To gain an insight into the structure of the HL7 message types, we analyzed the informal description of HL7 from [HLG94] and defined an object-oriented modeled for the data structure of HL7 messages [HK95]. This model can be used as the basis for the component schema of the corresponding component DBS, which could be specified using, e.g., the object definition language of ODMG-93 [Cat96]. The corresponding agent would intercept the HL7 messages from the subsystem and announce changes when they are detected. The forthcoming version 3 of HL7 will be accompanied with an object-oriented data model [RQ96]. This will simplify the task of wrapping HL7-messages.

4 Related work

In [RHC$^+$96], the techniques for federated DBSs are being deployed in the application domain of HISs, whereby the management of changes to the structure of federated DBSs is discussed. However, this approach does not discuss integrity control for replicated information.

Replicated data is employed in distributed DBSs to enhance data availability and performance: multiple copies of some data items are maintained, typically on separate sites, so that the data item can be retrieved even if some copies of the data item cannot be accessed due to system failures. However, this benefit of data availability is only realized at the cost of elaborate algorithms that hide the underlying complexity of maintaining multiple copies of a single data item. The difficulty lies in keeping the copies consistent with each other while at the same time maximizing the data availability and performance. The algorithms which address these problems are called *replica control algorithms* [BHG87].

With failures, however, writing all copies within a transaction can cause indefinite blocking, which is unacceptable in practice. Hence the write-all approach can be modified to write all copies available to the transaction coordinator. Unavailable replicas receive changes on a deferred basis. The most commonly known protocol of this genre is the *primary copy protocol.* A two-phased commit protocol is required to guarantee a consistent view of the replica (1-copy-serializability [BHG87]). To some extent, the basic principle of this protocol can be compared to our change algorithm with one master copy for data items, but we do not guarantee a consistent view of the replica since the changes are not executed in transaction boundaries. Also, for distributed DBSs it has been suggested to replace the 1-copy-serializability with, e.g., ϵ-serializability to allow asynchronous updates [PL91]. Temporary inconsistencies in replicas may be seen by queries with this asynchronous approach.

The integration of replicated information across autonomous subsystems within hospitals can only be achieved by weakening the autonomy requirements of component DBSs. Therefore, a way has to be found for introducing global integrity maintenance without restricting local autonomy too much. Our approach preserves a high degree of local autonomy by applying mechanisms of active databases on the global level of integrity maintenance through decoupling triggered and triggering transactions.

Decoupling of triggered and triggering transaction of change operations is reasonable in HISs, because therein predominantly insertions of new information occur: you rarely see outdated information that has been updated somewhere else, you only see new inserted information later on. Therefore, the weaker consistency is acceptable in this environment. Additionally, *lazy* replication algorithms that asynchronously propagate replica changes to other subsystems *after* the updating transaction commits are less deadlock prone than *eager* replication algorithms that propagate replica changes *before* the updating transaction commits, because the transactions have shorter duration [GHOS96].

5 Conclusions

A HIS is a complex *system of systems* which requires a well designed organization at the software architecture level. For digital information that is needed in hospitals, it is a major requirement to integrate the replicas of information about the same real world phenomenon which are stored in dissimilar and autonomous subsystems.

This paper presents our approach to federated integration of replicated information within hospitals. An architecture which is based on the reference architecture for federated database systems [SL90] and adapted to the specific demands on integration of replicated information is presented. This architecture is the basis for associated algorithms that restore the integrity of replicated information when changes occur. The change algorithms are based on the schema architecture. This approach keeps these algorithms simple and the analysis of the dependencies within the schema architecture can be used to detect possibly infinite loops of change propagation or deadlocks.

The schema architecture is extended to support change algorithms with one or multiple master copies for data items. Multiple master copies for data items should be avoided [SWG+92, GHOS96], but sometimes legacy systems have to be integrated which store the data in their own repositories. However, a federated architecture supports a smooth migration from legacy systems to modern information systems which do not require multiple master copies.

In contrast to the current state of the art in connecting subsystems within hospitals through communication servers, a tightly coupled federated DBS whose data integration is on the basis of schema integration is capable of supporting subsystems to interoperate. Instead of enabling the subsystems to send messages for information exchange, the exchange of information is accomplished through updates of replicated data in subsystems by the federation layer, which *knows* the dependencies among replicas. This approach allows to analyze and optimize the data flow within hospitals.

References

[AHW95] A. Aiken, J.M. Hellerstein, and J. Widom. Static analysis techniques for predicting the behavior of active database rules. *ACM Transactions on Database Systems*, 20(1):3–41, March 1995.

[BHG87] P.A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database systems.* Addison-Wesley, 1987.

[Cat96] R. Cattell, editor. *The Object Database Standard: ODMG-93, Release 1.2.* Morgan Kaufman, 1996.

[CT95] S. Conrad and C. Türker. Active Integrity Maintenance in Federated Database Systems. Preprint Nr. 9, ITI, University of Magdeburg, November 1995.

[Dat95] C.J. Date. *An introduction to database systems.* Addison-Wesley, 6th edition, 1995.

[EK91] F. Eliassen and R. Karlsen. Interoperability and object identity. *ACM SIGMOD Record*, 20(4):25–29, December 1991.

[ESP92] C.-Th. Ehlers, H. Schillings, and P.M. Pietrzyk. HIS and integration. In A.R. Bakker, C.-Th. Ehlers, J.R. Bryant, and W.E. Hammond, editors, *Hospital Information Systems: Scope – Design – Architecture*, pages 49–56. North-Holland, 1992.

[GHOS96] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. *SIGMOD Record*, 25(2):173–182, June 1996. (Proc. ACM SIGMOD International Conference on Management of Data).

[Ham93] W.E. Hammond. Health Level 7: A protocol for the interchange of healthcare data. In G.J.E. De Moor, C.J. McDonald, and J.N. van Goor, editors, *Progress in Standardization in Health Care Informatics*, pages 144–148. IOS Press, 1993.

[HK95] W. Hasselbring and A. Kröber. Requirements analysis on a configurable HL7-based communication server with OMT and executable models. In H. Krumm, editor, *Entwicklung und Management verteilter Anwendungssysteme*, pages 31–40, Münster, October 1995. Krehl-Verlag. (in German).

[HLG94] HL7 Group. Health level seven: an application protocol for electronic data exchange in healthcare environments, version 2.2. Technical report, Health Level Seven, Inc., Ann Arbor, USA, December 1994.

[KLB96] T. Kudrass, A. Loew, and A.P. Buchmann. Active object-relational mediators. In *Proc. First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, pages 228–239, Brussels, Belgium, June 1996. IEEE CS Press.

[LGM96] W.J. Labio and H. Garcia-Molina. Efficient snapshot differential algorithms for data warehousing. In *Proc. 22th International Conference on Very Large Data Bases*, pages 63–74, Bombay, India, September 1996. Morgan Kaufmann.

[McD95] C.J. McDonald. News on U.S. health informatics standards. *M.D. Computing*, 12(3):180–186, 1995.

[Mey88] B. Meyer. *Object-oriented Software Construction.* Prentice Hall, 1988.

[PBE95] E. Pitoura, O. Bukhres, and A. Elmagarmid. Object orientation in multidatabase systems. *ACM Computing Surveys*, 27(2):141–195, June 1995.

[PD95] H.U. Prokosch and J. Dudeck, editors. *Hospital Information Systems: Design and Development Characteristics; Impact and Future Architecture.* Elsevier, 1995.

[PL91] C. Pu and A. Leff. Replica control in distributed systems: an asynchronous approach. *ACM SIGMOD Record*, 20(2):377–386, June 1991.

[RSC97] Rational Software Corporation. The Unified Modeling Language. Documentation Set Version 1.0, Santa Clara, CA, January 1997. (available from `www.rational.com`).

[RHC+96] M. Roantree, P. Hickey, A. Crilly, J. Cardiff, and J. Murphy. Metadata modelling for healthcare applications in a federated database system. In O. Spaniol, C. Linnhoff-Popien, and B. Meyer, editors, *Trends in Distributed Systems: CORBA and Beyond, International Workshop TreDS '96*, volume 1161 of *Lecture Notes in Computer Science*, pages 71–83, Aachen, Germany, October 1996. Springer-Verlag.

[RQ96] W. Rishel and J. Quinn. Software components, the clinical workstation and healthcare networks: How HL7 is helping you get there. In *Proc. Healthcare Information and Management Systems Society's Annual Conference*, Atlanta, Georgia, March 1996.

[RS95] E. Radeke and M.H. Scholl. Functionality for object migration among distributed, heterogeneous, autonomous database systems. In *Proc. 5th International Workshop on Research Issues in Data Engineering: Distributed Object Management (RIDE-DOM '95)*, pages 58–66, Taipei, Taiwan, March 1995. IEEE Computer Society Press.

[SL90] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SPFW90] E.H. Shortliffe, L.E. Perreault, L.M. Fagan, and G. Wiederhold, editors. *Medical informatics: computer applications in health care.* Addison-Wesley, 1990.

[SS95] I. Schmitt and G. Saake. Managing Object Identity in Federated Database Systems. In M. Papazoglou, editor, *Proc. 14th International Conference on Object-Oriented and Entity-Relationship Modeling (OOER'95)*, volume 1021 of *Lecture Notes in Computer Science*, pages 400–411, Gold Coast, Australia, 1995. Springer-Verlag.

[SWG+92] W.W. Stead, G. Wiederhold, R. Gardner, W.E. Hammond, and D. Margolies. Database systems for computer-based patient records. In M.J. Ball and M.F. Collen, editors, *Aspects of the computer-based patient record*, pages 83–98. Springer-Verlag, 1992.

[WC96] J. Widom and S. Ceri, editors. *Active Database Systems – Triggers and Rules For Advanced Database Processing.* Morgan Kaufmann Publishers, San Francisco, 1996.