Global DB Views in a Federation of Autonomous DBS

J. M. V. de Castilho[1]
Instituto de Informatica – UFRGS
Porto Alegre, Brazil

R. P. da Rocha
Instituto de Informatica – UFRGS
Porto Alegre, Brazil
rrocha@inf.ufrgs.br

T. Härder
Computer Science Department
University of Kaiserslautern, Germany
haerder@informatik.uni-kl.de

J. Thomas
UBS AG
Basel, Switzerland
joachim.thomas@ubs.com

Abstract *The Institute of Computer Science of the Federal University of Rio Grande do Sul is participating in a project to develop a computer system infrastructure for supporting the Brazilian health ministry's SUS (Sistema Unificado de Saude) program. In order to construct this heterogeneous system, we have to start from existing information sources (possibly "legacy") and their services and gradually integrate them with new software and hardware architectures, guided by requirements evolution and technical innovations. This requires to combine distributed query processing with a service-based middleware framework as defined, e.g., by CORBA. This paper describes the conceptual and implementational aspects of the SUS-specific solution to this challenge.*

Keywords: *Heterogeneity, Database Federation, Health Information System, Schema Integration, Middleware, Service Distribution*

1 Introduction

The Brazilian Health Ministry has the SUS (Sistema Unificado de Saude) program, in which all member hospitals are authorised and paid to assist patients of the National Institute of Social Security (INSS). The control of the SUS is being moved from the federal level to the municipal level, and the cities now face challenges such as offering better services to the patients and physicians as well as better managing of expenses.

The data processing company of the City of Porto Alegre (PROCEMPA), in partnership with the Institute of Computer Science of the Federal University of Rio Grande do Sul (II/UFRGS) and other research centres in computer science and medical computer science, have started a joint project named SIDI (Sistemas de Informação Distribuídos e Inteligentes) [3] to develop a computer system that integrates all autonomous local databases of the member hospitals and procedures for controlling hospitals and patients in a unified way. This subject is also being considered as one of the topics for study under the DoCAIA co-operation program between the University of Kaiserslautern and II/UFRGS.

[1]Prof. de Castilho passed away while this paper was in the review process.

The health information system that is being designed in the scope of SIDI (called HIS hereafter) has as general objectives the control of data and use of resources of a communal public health system. It is a distributed computing system, and the main sites of the corresponding network are the hospitals and other health care units associated with the public health system.

Considering the medical and practical viewpoint of the distributed system, we identify the following classes of users (and of uses) for the system:

- Physicians and other health staff having direct contact with patients and recording their clinical data. These users normally access the hospital's local information system, entering or querying patients' local data. They may use HIS whenever they need universal access to health records that are stored elsewhere in the network, for instance, in another hospital's database, where the patient being examined has already been assisted in some way, and where data on him/her is available. Those data may serve to help physicians in their diagnosis to complement the patient's record currently being collected.
- Local hospital administrative staff, who access HIS for controlling unnecessary expenses for patients. The system may be used to collect account information of the most recent visits of a patient in the whole health system to prevent him/her having repeated consultations, exams, and treatments in more than one hospital at public expense.
- SUS administrative staff, who access the system for accounting purposes, calculating the monthly values that the municipality should pay to each hospital, to cover staff expenses and other resources used during the month. These calculations are not of a simple nature, and are subject to strict regulations imposed by governmental authorities. One may also consider the possibility of frauds, and that some data supplied by hospitals may not be correct. Regulations may also change with exasperating frequency.
- Health care researchers, who need to access data of local systems in a unified way to support their medical researches, the discovery of extraordinary situations and epidemics, and administrative decision makers who look for these integrated data for planning improvements in the public health system, corrections of the use of available resources, frauds detection, and in taking other necessary administrative decisions. These tasks could greatly benefit from providing a so-called "data warehouse" keeping the required aggregate data and summaries readily and efficiently accessible.

These usage scenarios should give some insight into the requirements for a federated system. From a technical viewpoint, the system being designed should be integrated, as tightly as possible, with the already existing, local information systems. New modules have to be installed at each participating site for external access to local data. For this reason, we need a heterogeneous system with global and local functionality whose system architecture will reflect many aspects of federated databases [19].

The HIS global functionality involves a unified control of patients, their diagnoses and treatment procedures (e.g., exams and examinations). This concerns providing global integrated views of these entities based on data supplied by local information systems. For end-users of the HIS, Web browsers are made available, and queries that are considered important for the work of physicians and the HIS staff are predefined as parameterised queries and presented to users in form-based Web pages. Moreover, the user-interface offers a special Web page which enables the specification of ad-hoc queries, e.g., in SQL.

The HIS global functionality is carried out by decomposing queries addressed to the global view into subqueries which are evaluated at the local sites. After having received the results of the subqueries, HIS combines them to produce the complete query result. This involves merging entities such as *patient* which might occur in more than one local site, thus potentially giving way to all kinds of problems associated to redundant, inconsistent, or incomplete data.

In order to homogenise the requirements HIS poses to local hospital information systems, we must define the local functionality that has to be provided by each local site. These functionality concerns local information on global entities (i.e., patients, diagnoses, or procedures) as well as capabilities to query these entities. One important aspect of HIS is to respect the autonomy of the local systems since they will continue their regular processing.

To support the above scenario appropriately, the health information system will be designed in such a fashion that it integrates existing, potentially heterogeneous information sources and usage patterns. They constitute the foundation on which new, globally accessible services will be built. In order to cope with changing functional requirements, evolution and extensibility must also be accounted for when laying out the system architecture.

Other important technical aspects of the HIS federation concern data access, transmission, security, and access permissions. For those features, HIS adopts the CORBA middleware architecture and specifications. A crucial additional requirement is data warehousing which will, however, not be treated by the initial solution.

In order to construct this heterogeneous system, we have to start from existing information sources (possibly "legacy") and integrate them with new software and hardware architectures, in a gradual process guided by requirements evolution and technical innovations. This poses two fundamental problems: extensibility and evolution - in the initial phase of establishing the system as well as in later phases where local sites and global services may evolve separately and at different speeds.

Altogether, basically three problem fields must be accounted for in the development of HIS:

- data model issues related to the schema architecture of the federated system and the corresponding mapping/homogenising of data,
- operational issues related to constructing and providing services at different levels (local/global) and to integrating them in an overall system architecture,
- possible evolution and extensibility scenarios for HIS.

In accordance to these topics, this paper is organised as follows. After a technical discussion of general system aspects (Sect. 2), Sect. 3 evaluates the use of a multi-level schema architecture which provides a homogenised, global schema, a prerequisite for flexible accessibility of data managed by local sites. Sect. 4 will discuss operational issues of the underlying federated system architecture. The subsequent section then sketches future potential evolution paths from the initially targeted solution, before we summarise and conclude this paper in Sect. 6.

2 General System Aspects

HIS is supposed to integrate up to thirty hospitals and small health care units as well as other organisations and administrative facilities related to health care. Many of these institutions operate local database systems. Hence, establishing HIS fundamentally means to construct a federation of these sites [19].

2.1 Heterogeneity Issues

As motivated in the introduction, HIS will consist of a core implementation providing new functionality that is complemented by and built on top of existing, autonomous local systems. Although efforts are made towards uniform data modelling and management in HIS, a federation consisting of a collection of autonomous and decentralised information sources can never be designed as a homogeneous distributed DBS. Moreover, new demands cannot be anticipated in general and surely will exacerbate overall system complexity. For example, a local hospital needs to introduce some new data type for storing information related to a novel treatment recently installed. Another source of system divergence might be independent maintenance and development of system components, as, e.g., DBMS engines. System design therefore must cope with various kinds of heterogeneity, both concerning hardware as well as software.

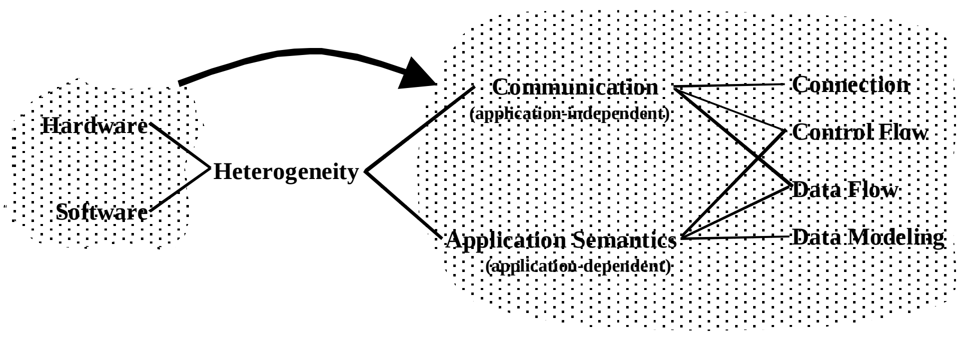

Figure 1: Different levels of heterogeneity

For the scope of our current discussions, heterogeneity of hardware and software can be reduced to heterogeneity of communication and of application semantics (cf., Fig. 1). This simplification is justified since implementing HIS means to establish information exchange among previously detached sites that is semantically meaningful to the participants.

Heterogeneity of communication calls for application-independent infrastructure to establish connections between information sites and to enable flow of control and data among them. On the other hand, the various information sites participating in HIS pose different application-specific requirements to the flows of control and data. The same holds for all aspects of data modelling. Starting with a common system architecture, we will subsequently discuss both problem spheres.

2.2 Common System Architecture

Because of the variety of data sources to be integrated, and due to the likely evolution of HIS, heterogeneity at the level of communication can be best handled by distributed-object middleware, as defined by CORBA [18]. In Sect. 4 we will elaborate in detail how CORBA and associated technologies are going to be employed in HIS. For the time being, we will briefly discuss the advantages of CORBA for the implementation of our target system.

The CORBA approach provides a single, consistent view on distributed computing. The OMG has developed a basic object model and a reference architecture (OMA, Object Management Architecture) upon which applications can be constructed. OMG's object model provides a common object semantics that is the basis for application-specific object definitions. It also marks the cornerstones of data flow in OMA (objects and their services, messages and their result values). The OMA itself defines various facilities necessary for distributed object-oriented computing. It is made up of four constituents.

- The *Object Request Broker* (ORB) is a common communication bus for objects. It fosters a homogeneous setting for interacting sites in HIS.
- *Object Services* provide fundamental (infrastructure-level) object interfaces necessary for building object-oriented distributed applications. The Object Query Service and the Security Service are of particular interest. The previous service provides a uniform interface for querying and handling query responses, the latter permits to establish a uniform sphere of access control, involving authentication and identification of clients and servers.
- *Common Facilities* (offering standardised access to common application interfaces) as well as *Application Objects* (to integrate independently developed application interfaces) will not be considered for the scope of this paper.

CORBA-based technology provides a solid environment for tackling many of the heterogeneity issues in HIS. However, providing a core object model does not yet resolve any problems stemming from heterogeneity at the level of application semantics. This aspect will be considered subsequently.

2.3 Autonomy Issues

Heterogeneity of application semantics must be bridged in order to be able to define and establish global functionality in HIS. All central facilities must be founded on common notions (data items, operations, naming conventions), yet must still be interpretable in local contexts.

These requirements call for a balance between local autonomy and integration. Initially, HIS will necessarily have a high degree of local autonomy. This is due to the fact that HIS is formed by individual (local) DBS exhibiting a high degree of autonomy. Many decisions and measures are taken locally, as, for example, all aspects of system administration, daily operation, and long-term evolution of related services. In particular, specifying and optimising the internal DB schema can only be accomplished by the local organisations, since their specific configuration as well as performance requirements have to be respected. Thus, the design of a DBS federation has to explicitly regard these system properties. In particular, all problems of DB schema design and evolution, system growth, application development, etc., are solved locally.

However, some agreement and uniformity has to be achieved among the participating local DBS in order to allow for the required co-operation. A prime issue is a kind of equivalence of the local conceptual DB schemas - at least in their kernel parts, as discussed in Sect. 3. Otherwise, DB schema mapping as well as semantic interpretation of metadata and user data would pose insurmountable problems. Moreover, data security/ privacy requirements have to be solved in a uniform way permitting suitable access for applications of the global DBS in the HIS federation [4, 17].

3 Data Model Issues

So far, we have described the DBS federation and its environment from an abstract point of view. In particular, we have discussed the reasons of incorporating at least some kind of heterogeneity into system design. We will refine the key issues related to DB schema mapping in this section. For this purpose, we will outline the schema architecture for the DBS federation, introduce the concept of a Minimum Conceptual Schema (MCS), and discuss the problems of including data of autonomous DBS in a joint (global) view.

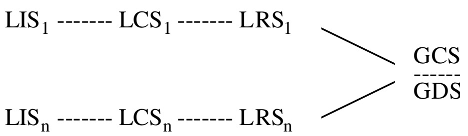

Figure 2: General schema architecture for DBS federations

3.1 Schema Architecture for the DBS Federation

In order to process data at the local and global levels of a DBS federation, different description models for data have to be specified. These models represent a complete definition of the data to be processed and viewed at the resp. level. Hence, they must describe structure, integrity constraints, operations, as well as access control. On the other hand, these models serve as source/target descriptions of the mapping process to be applied when data is selected from a local DBS for its integration into a global DBS view (and vice versa). For these reasons, a general schema architecture was proposed for DBS federations [19, 12]. As illustrated in Fig. 2, a multi-level architecture is needed to govern the required schema mapping process.

A Local Internal Schema (LIS_i) contains the description of the physical DB representation which is provided to store and retrieve data in a local DB. In order to derive a logical object representation from the local DB_i, information of the Local Conceptual Schema (LCS_i) has to be used. Since the local data models may exhibit various differences, an additional description layer is added to homogenise the participating models and to offer a unified data description to the DBS federation. Hence, the key to cope with heterogeneous data models is the introduction of a Local Representation Schema (LRS_i) and the corresponding LCS_i-LRS_i mapping. The Global Conceptual Schema (GCS) contains the description of the global DBS in the federation. A special Global Distribution Schema (GDS) is required to provide additional information for the GDS-LRS_i mapping. GDS may contain location-dependent predicates to keep track of the distribution of user data. For example, if specific X-ray information is collected only by certain hospitals, such predicates could be used for selective search to optimise query evaluation in the DBS federation.

3.2 Minimal Conceptual Schema

It is not sufficient to provide a homogeneous schema (GCS) and the structural mappings to the local ones and vice versa (LCS_i). Moreover, uniform meaning and treatment of values have to be guaranteed. For this reason, a so-called Minimal Conceptual Schema (MCS) is defined for each local DBS.

The minimal conceptual schema has been designed to contain the most relevant information for the HIS administration and for the medical community and is considered to be the core part of each local DB Schema (LRS_i) as well as of the global schema (GCS). This common schema is defined by a central agency in cooperation with hospitals and other cities, and is structured in a way that permits unified and centralised control of hospitals and patients. The medical community, however, demands complete, detailed, and structured information of the patients' examinations, involving all hospitals of the network. Physicians may desire to retrieve the records of a patient for supporting their diagnoses, as well as to query the global database for helping in their research, diagnosis, and patient treatments.

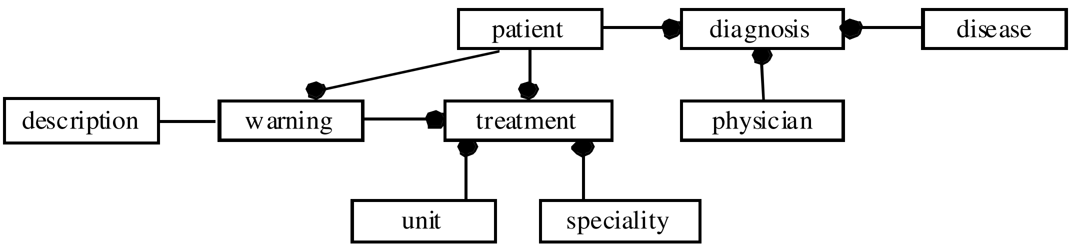

Figure 3: Minimal conceptual schema

Summing up, the MCS is basically composed of objects such as physicians, health units, patients, treatment, diagnoses, diseases, procedures, and warnings, as presented in Fig. 3. A *diagnosis* describes the results of an examination realised by a *physician* on a *patient*, and may have a *disease* as conclusion. A *treatment* identifies a medical activity performed for a patient, such as an examination, exam or even an hospitalisation. Each treatment is related to a procedure, that classifies it for billing and statistical purposes. Patient particularities, such as allergies, blood type, or vaccines, that are important to be alerted before starting any medical procedures are described by *warnings*.

The MCS has to be powerful enough to satisfy expectations of HIS users as well as being as close as possible to the least common denominator of all local data representations to avoid inconsistencies. Globally represented entities and relationships must be understood by all local DBS representations. Fortunately, local systems have some common data representations and values, due to their being obliged to report to the Health Care Ministry. These reports, however, only present procedures and diseases, since they were designed for accounting purposes as well as for helping decisions makers on health care policy. There is no regulation that enforces hospitals to keep data on patients and their relationships to treatments and diagnoses. Nevertheless such information is present in most local DBSs in some form.

Nowadays, we observe a crescent interest in standardising health care objects and values due to the necessity of exchanging such kind of information. Obviously, the MCS should reflect these standardisation efforts in order to provide universal value as well as to facilitate the integration of local DBSs that adopt those standards. Since the standardisation efforts of health objects are concentrated on patient records, the MCS describes a patient containing essential attributes which are adopted from the HL7 [16] and CorbaMed [9] standards. These attributes define the patient profile, which, when combined, allow her/his identification (e.g., name, birth date and place, identification card number, mother's name). Moreover, a patient record contains additional non-standard attributes such as contact person, address, civil state, etc.

Another crucial problem concerns attribute values, since in autonomous systems there may be no global control over their specification and usage. Hence, synonyms, homonyms and other kinds of ambiguous descriptions may occur [15, 13]. Fortunately, we can rely on standardisation efforts that classify diseases and procedures [2], which are adopted by many hospitals and by the Health Care Ministry. The MCS takes this direction, adopting standards and widely accepted codes and names for diseases and procedures. This homogenisation enables precise statistical applications over the MCS, since disease and procedures classify treatments and diagnoses, respectively.

3.3 Mapping Aspects of the Federated System

In the schema architecture (Sect. 3.2), the GCS describes the global DBS whose objects are generated by LRS_i-GCS mappings from their persistent states in local DBSs. These mappings concentrate on combining objects supplied by the local DBSs. The location aspects of these objects may significantly facilitate this process and may result in different approaches for defining global identification for objects, as well as for generating their integrated views.

Types of object distribution in a federation	Definition	Strategy to generate/query the integrated version of the object
Overlapping inconsistent	objects that occur repeatedly in more than one local DBS with different attribute values	merging object versions from the local DBSs, providing strategies for handling the inconsistencies
Overlapping consistent	objects that occur repeatedly in more than one local DBS with identical attribute values	union of local DBSs' object versions followed by elimination of duplicates
non-overlapping	objects that occur in only one local DBS	union of local DBSs' object versions

Table 1: Types of object distribution in a federation

Analysing the MCS (which defines GCS and LCS_i) and the distribution of its objects in the federation of hospitals, we observe that some global DBS objects may overlap in multiple local DBSs (e.g., *patient*, *physician*, *disease*), while others definitely occur in one local DBS only (e.g., *treatment* and *diagnosis*). Furthermore, some overlapping objects may have identical attribute values in all local occurrences (e.g., diseases whose instances are defined by an international classification), while others may have different values for identical attributes (e.g., *patient*), requiring additional processing for handling inconsistencies.

Summing up, we classify global objects in non-overlapping, overlapping consistent and overlapping inconsistent (Table 1). From a global query processing point of view, non-overlapping objects may be provided by a union of objects supplied by the local DBSs. In the case of overlapping consistent objects, the union has to be followed by a duplicate elimination operation. However, overlapping inconsistent objects require for a merging operation, involving their local DBS versions, to handle their inconsistencies. A more detailed elaboration of these aspects can be found in [8].

This classification also impacts the construction and maintenance of global identifiers in the federated system. It is crucial to the mapping system to define global identifiers for objects that indicate how these objects are generated from persistent data in local DBSs [11].

In our federation, we have objects which are locally identified by a universal key (e.g., physicians, who are identified by their register in the medical council). In these cases, we may use the universal key to identify the object in the global DBS, and the global object may be generated by querying its persistent states in local DBSs using this key. We have also cases of non-universally identified objects that, however, may be globally identified by a composite key (e.g., treatments and diagnoses). In these cases, the composite key is formed by the local keys of the object combined with the identification of the local DBS. This simplifies the generation of global objects, that may be performed by querying the local DBS identified by the composite key, using their local identification, also extracted from this key, as selection criterion.

The most complex case of object identification occurs for patients. The Brazilian health care policy establishes that every person be assisted by the SUS, independent of any document of identification, social security number, or even a SUS card (if the patient lives in a city that provides such a service). Hence, we have no standard identification for patients in local DBSs. This means we need a service for patient identification, which can compare local patient profiles and identify which of them represent the same global object, providing a global identifier to the integrated version of these objects.

Object Identification	**Definition**	**Strategy for Naming Service**
Composite key	non-overlapping objects that may be identified by local keys composed with identifier of local DBS	local keys + ID of local DBS
Universal key	overlapping or non-overlapping objects that have a unique identification key in the federation	universal key
Profile	overlapping objects that may be identified through their properties (profile)	map local object profiles (including identifiers) to a global identifier

Table 2: Types of object identification in a federation

The mapping system employs an object identification service that guarantees the immutability and singularity of object identifiers. It is used by the mapping system to provide global object identifiers for objects that are being generated from their persistent states in local DBSs. Furthermore, it also provide the reverse operation, that is, the object identifiers of the local objects that compose a global object. This permits to provide uniform implementation-independent interfaces for generating MCS identifiers. Summing up, Table 2 presents the types of object identification in our federation and the strategies for generating the global identifier for the integrated version of these objects.

In the next section, we provide an extensible architecture capable of incorporating the services and functions discussed so far.

4 Operational Issues of the Federated System

As motivated in the previous sections, a suitable approach to HIS must provide

- platform-independence, in order to seamlessly integrate heterogeneous systems,

- flexibility, for being able to cope with the particularities of participating sites,
- evolvability, so as to favour the future development of HIS and to meet new demands to the system.

In order to fulfil these requirements we intend to integrate the local constituents of HIS, i. e., loosely-coupled DBSs and their set-oriented query facilities, with a service-based middleware architecture according to the CORBA standard. In Sect. 4.1 we will discuss basic considerations of query processing in HIS. Thereafter, Sect. 4.2 will sketch optimisation measures for querying a federation of loosely-coupled DBSs. Sect. 4.3 will concentrate on the main services required by HIS, and Sect. 4.4 presents how these services are distributed using CORBA as platform. In Sect. 4.5, we will outline a suitable implementation approach for the query service.

4.1 Basic Aspects of Query Processing

Query processing in HIS is oriented at the layered schema architecture (Sect. 3.1) in which local and global functionality perform data and query translations from one schema layer to another. Queries addressed to the global DBS are decomposed into subqueries which are evaluated in the local DBSs. Subqueries are expressed in the same query language as the original query, since GCS and LRS have an identical data model. In a local DBS, a subquery is translated to the given query language (dialect) referring to the specific LCS_i. These subqueries are performed by local DBSs, and their results are translated to LRS format. After having received the results of the subqueries, the global DBS combines them to produce the complete query result.

By sending several subqueries to the participating local DBSs at a time, query evaluation runs in parallel as far as possible. For simple queries, the task left to the global DBS is to assemble the partial results by union or merge operations. However, more complex queries require join operations in the global DBS. Such queries have high processing costs, even more so because potentially many data have to be transferred from a local DBS to the global DBS to be joined.

4.2 Optimisation Issues in Query Processing

As already mentioned, data location aspects of global objects may determine simplifications in their mapping operations in the global DBS: non-overlapping and overlapping consistent objects may be generated by a simple UNION operation of the objects supplied by local DBSs, contrasting to overlapping inconsistent objects that require complex merging operations to handle inconsistencies.

The data locations may also determine other strategies for query processing. For example, if we have, in each local DBS, a complete class extension of overlapping consistent objects (i.e., identical copies of a class extension A_0 in all local DBSs), we may push down joins to local DBSs. This optimisation is based on the following equivalence rule, considering that B_1 and B_2 represent incomplete extensions of class B in the sites 1 and 2, respectively:

$$(B_1 \text{ UNION } B_2) \text{ JOIN } A_0 \Leftrightarrow (A_0 \text{ JOIN } B_1) \text{ UNION } (A_0 \text{ JOIN } B_2).$$

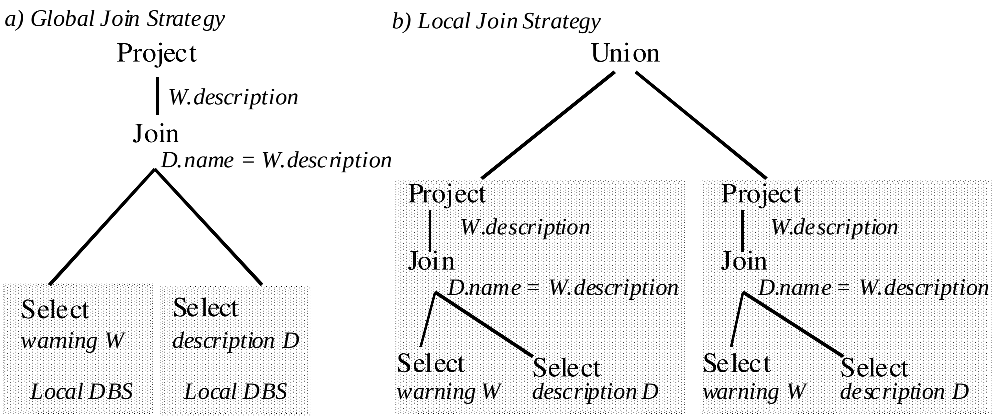

Figure 4: Global join strategy vs. local join strategy

This optimisation is possible, e.g., in the case of diseases, since each local DBS has the complete international classification of diseases. For example, in the query "Select warnings whose descriptions contain a disease name", each local DBS could perform the join between warnings and description of diseases. Fig. 4

contrasts global and local join processing strategies for this query.

Furthermore, some overlapping inconsistent objects with universal key may be converted to consistent objects with complete coverage of instances in all local DBSs. Thus, we may perform joins involving those objects locally, and we may avoid complex merging operations at the global level. For example, we could propagate to the local sites a complete list of physicians, which could be maintained by the Medical Council.

Another important issue of query processing is to find out whether a (sub-) query has to be sent to a specific database system, to a group of DBSs, or even to all local DBSs. For this purpose, GCS and GDS should provide location-sensitive information. For example, queries about specific kinds of diseases can be decomposed into subqueries addressed only to DBSs of those local sites which treat this kind of disease. This requires to analyse context information characterising the local DBSs.

Global queries involving joins on relationships may also be optimised by pushing these joins down to local DBSs, since global relationships reflect local relationships. In the GDS-LRS_i mapping, a LRS relationship among local versions of GDS objects is mapped to a GDS relationship among the integrated versions of these objects. Hence, a global query on a relationship may be processed in such a way that the global DBS requires, from local DBSs, not only the objects, but also their relationships. Thus, it only has to generate integrated versions of these objects and transform the relationships to refer to these integrated objects.

For example, the query "Select patients born in Porto Alegre and their treatments" could have an evaluation strategy where patients and treatments are related locally, and the resulting joined objects are sent to the global DBS that unifies them. Fig. 5 contrasts this approach with a traditional one that joins patients and treatments at a global level.

As mentioned above, constructing a complete ad-hoc query processing system could be very expensive and time-consuming for the first version of HIS. A plausible alternative is to initially define a system that only processes the most relevant queries for the SUS administration, and then evolve to a complete ad-hoc query processor later on. To achieve this objective, we exploit the fact that the majority of HIS queries is simple (i.e., does not involve expensive global joins), performing selections over relationships (providing optimisation potential as described above). For example, physicians and administrative staff typically require on-line queries for selecting patients, their treatments, and diagnoses in order to make their prescriptions, authorisations, and payments.

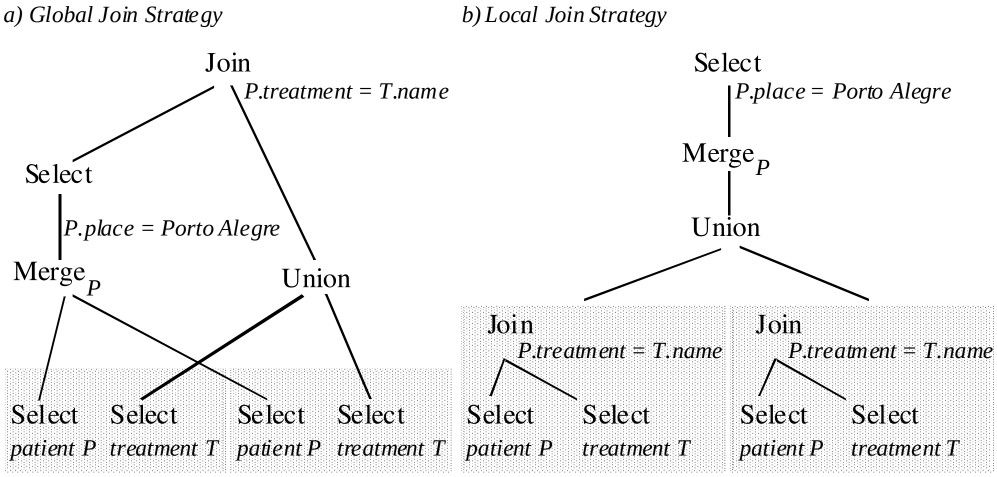

Figure 5: Pushing down join on relationship

As a consequence, the first version of the query processor will only execute simple queries, dispatching the simple queries to local DBSs, and unifying the results, performing the generation of the global objects. At a subsequent stage of system development, query processing will be extended to deal with more sophisticated queries, involving complex associations of information such as for discovering frauds, epidemics, and excessive expenses for treatments, as well as for supporting decisions. These queries require complex mechanisms for defining their processing strategies [7], since global queries have to be detached to identify which parts will be sent to local DBS, and which parts will remain in the global DBS.

4.3 Service-Based Approach

HIS functionality will be provided through services, whose design is based on the CORBA standard [18]. A service is specified at a high abstraction level, which defines interfaces to service components (using CORBA Interface Definition Language - IDL) and their interplay. Furthermore, HIS adopts OMA services, such as querying and security. This avoids redundant specifications, and promotes standardisation and interoperability.

This service-based approach allows service implementations to be initially simple and limited and evolve to more complete versions later. A service evolution does not need to bear huge consequences to the overall system architecture, since CORBA strictly separates interfaces and their implementations. Furthermore, services may evolve at different speeds and times. For example, we may have a complete query system interplaying with an inoperative cache [1], as well as a complete cache operating with a simple query system.

This approach also permits that service implementations could be reused or extended, giving great flexibility to the federation to incorporate new local DBSs. This occurs, for example, when integrating a new local DBS into the federation, which has many similarities with another local DBS already participating in the federation.

Based on the service paradigm, HIS is composed of services such as *domain*, *identification*, and *mapping* which co-operate to perform schema translations in HIS layered schema architecture (Sect.3.1) These services provide modules of integration that occur locally, to perform LCS_i-LRS_i translations; and globally, to generate the GCS from LRSs.

Such a local module of integration is responsible for managing a target schema object extension (e.g., LRS_i in the LCS_i-LRS_i integration module) in a schema translation process. It controls the activation of schema object instances, avoiding their repeated occurrences and guaranteeing immutability and singularity of their identifiers. This service uses a *caching* service to store activated objects, and an *identification* service to provide immutability and singularity of their references. Thus, multiple references to an object denote the same activated object, and an object will always receive the same reference, when activated.

The functionality of these services may vary according to the kind of schema transformation they perform (LCS_i-LRS_i or LRS_i-GCS), as well as query processing capabilities. For example, since LRSs are placed in the intermediate layer (Section 3.1), and considering their unique client being the LRS_i-GCS schema translator, the cache functionality of the corresponding *domain* service could be eliminated. Thus, this layer would only perform object translations, transferring queried objects to GCS, without storing them.

In addition to the services mentioned before, HIS requires a global *query* service that perform the functionality outlined in Sect. 4.2. This service receives a query on target schema, decomposes it into subqueries to be executed in source schemas, collects their results, and produces the queried objects. The *query* service interacts with *mapping* and *domain* services to generate and activate the resulting objects. The *mapping* service isolates schema integration methods from other (integration) services. It supplies the subqueries to be sent to source schemas, and also provides post-processing operations, which will be used to generate the target schema objects from these subqueries' results.

4.4 Distributing HIS Services

CORBA provides implementation-language and location transparency for HIS local/global services. Interfaces of HIS services are specified in IDL and mapped to a CORBA-supported implementation language (such as C, C++, or Java) which is used by each global/local system component. To support design and implementation reuse of local services, an object-oriented framework [10, 20] is developed in Java, having its main classes resulting from mappings of IDL service interfaces. In this framework, abstract methods identify service functionality that depends on local system particularities (such as mappings), and therefore must be implemented by each local system.

In CORBA, object location transparency is provided by proxies, which represent the object in each remote place that it is being referenced. This proxy-based approach performs remote object invocation by reference, since an object implementation remains in its original site, and CORBA provides the dispatching of a message invoked on a proxy to its remote object implementation, and returns its response to the proxy.

In HIS, global services invoke local services through proxies. Furthermore, HIS GCS objects are described in IDL, so that clients can access them through proxies. For example, a HIS client having a proxy of the HIS query interface (standardised by OMA Query Service) may construct a query and submit it to HIS. As response, HIS returns a proxy of the queried objects collection, and iterators provide a way to handle its contents (proxies of GCS objects).

Similarly, the global system submits queries to local systems through proxies of local query services. The resulting LRS objects, however, are not accessed through proxies by global services, since it would be time- consuming and would overload CORBA. Instead, queried local objects are transmitted from local query services to the global service (called invocation by value). This object transferring is carried out by the middleware framework dLIMIT, as presented in the next section.

Hence, the *query* service introduces distributed query processing facilities into the middleware framework of HIS. Since the associated service implementation is potentially very complex, we will have a closer look at an implementation approach for the *query* service in the following subsection.

4.5 Implementing the Query Service

The middleware framework dLIMIT [14] provides a basis for building loosely-coupled DBS federations across the internet. dLIMIT offers a simple but powerful application programming interface (API), providing location and communication transparency. Thus, it facilitates the access of autonomous, heterogeneous information systems. Additionally, dLIMIT supports different types of access to underlying DBSs, and allows the encapsulation of these systems, e g., for security reasons. dLIMIT, based on Java [5] to achieve platform independence and to support the easy installation and maintenance of applications, is highly configurable. Thus, it is adaptable to different kinds of applications and their environments. dLIMIT supports both the use in Java applets and applications so that browser-based as well as stand-alone applications can be established [6].

dLIMIT performs distributed execution of global queries, which are treated as tasks. The dLIMIT client interface (Fig. 6) receives a task and passes it to the Distributor/Merger component, which is responsible for routing the task to its final destinations (i.e., source databases). This component interacts with Task Handlers, which are responsible for converting the task to each system-dependent query of the source databases. The Task Handler is also responsible for assembling the result objects returned from each source database into an AnswerSet structure. All answer sets provided by Task Handlers are included into a single answer set by the Distributor/Merger and passed back to dLIMIT clients. dLIMIT also specifies a pipelined functionality for answer sets, so that their elements may be collected at the time they are generated by their sources. Thus, the Distributor/Merger may populate a task result AnswerSet by creating threads which are responsible for polling each Task Handler AnswerSet. The task result AnswerSet is completed when all polled Task Handler answer sets signal end-of-output.

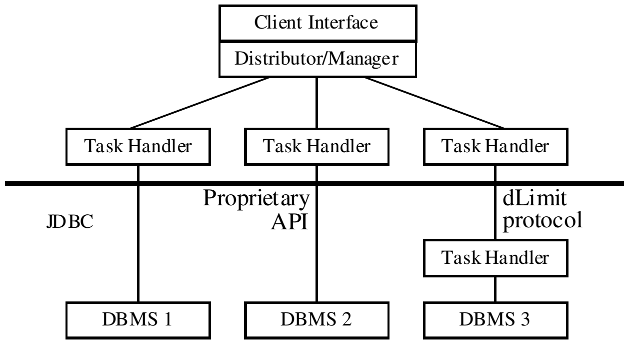

Figure 6: Architecture of dLIMIT

Task Handlers and answer sets provide standard interfaces that isolate source databases data access protocols (e.g., JDBC), and their distribution aspects. For task handlers that cannot access their source databases remotely, dLIMIT also allows them to be placed on the same server as their source databases.

Hence, in the context of HIS, dLIMIT can be used to perform task distribution and result collection between the global and local DBSs, i.e., between LRS_i-GCS and LCS_i-LRS_i schema translation modules (Fig. 7). Thus, a global LRS_i-GCS schema translator module uses the dLIMIT Distributor/Manager as middleware service for submitting queries to local DBS and collecting their results. HIS local modules (i.e., LCS_i-LRS_i schema translation modules) interact with the global module using the dLIMIT remote Task Handler data access protocol for transferring their objects.

Summing up, the main advantage of using dLIMIT for implementing the *query* service is that it not only offers location transparency and object transferring, but also performs middleware activities typical of federated DBSs such as splitting queries to be executed in local DBS, and collecting their results. Furthermore, dLIMIT is developed in Java and adopts well to the object-oriented setting implied by the CORBA framework chosen for HIS.

5 Evolution Scenarios

In the introduction, we have motivated the approach to start building up a HIS by linking and integrating existing local information sources and applications. Obviously, the resulting solution can only meet the essential needs of the SUS program. Future evolution of the system is therefore indispensable. In the following, we will sketch a likely evolution path for HIS aiming at improving system performance by integrating more tightly the distributed system architecture of HIS.

The initial system architecture of HIS is highly decentralised with only a small amount of integrative glue provided by the schema mapping and global functionality of HIS. Potential drawbacks of this initial solution are performance problems as well as redundancies of services.

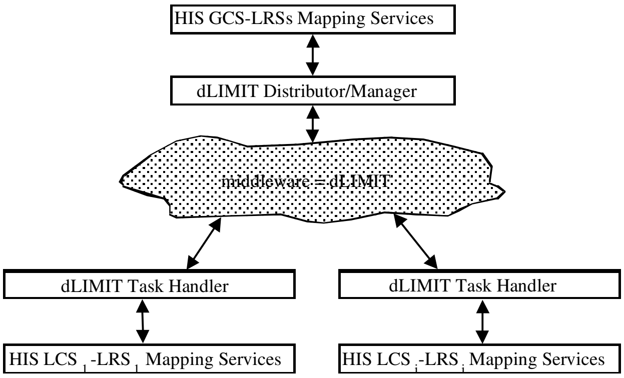

Figure 7: dLIMIT as HIS middleware

Performance problems may arise for various reasons. For example, a global service may have initially been implemented by simply mediating a clients' request to a local system. While this solution warrants short response times for local requests, the "path length" for global requests might incur severe performance penalties. Depending on the frequency of local vs. global inquiries, it might therefore be beneficial to lift the local service to a global level.

Redundancies of services is the second major disadvantage of HIS as targeted for the initial version. As mentioned before, the local systems participating in HIS meet the specific requirements of distinct groups of users (e.g., physicians of a certain hospital, administrative staff, etc.). While this implies that those systems are quite heterogeneous, it also insinuates that there might be comparable services implemented redundantly at various sites. The most prominent example of such a service is the identification of patients. As outlined in Sect. 3.3, HIS needs to provide global IDs based on locally generated ones. In the long run, however, a single, system-wide identification service should be established.

Summing up, achieving a tighter integration of HIS and its constituents is one desirable evolution direction of the system. To pursue this endeavour it is indispensable to align evolutionary measures on local systems with this overall global objective. To this end, location transparency as well as the proper distinction between interface definitions and their implementations are important benefits of the CORBA approach. Thus, the implementation of relevant global services might gradually be positioned at a global level without affecting existing global usage. However, such migrations might strongly impact local usage which are potentially rooted in legacy applications that do not provide any location transparency. How to minimise local modification efforts can only be decided on a case-by-case basis.

6 Summary and Conclusions

The primary focus of our article has been on the derivation of a suitable DB schema architecture enabling the access of heterogeneous and autonomous local DBSs. This aspect concerns the unification/convergence of syntactical and semantic discrepancies and divergences for attributes and values; the concept of the so-called Minimal Conceptual Schema was proposed to alleviate the related problems. We also have identified mapping issues of MCS such as the location aspects of MCS entities in the federation and how they can be globally and locally identified. As a consequence, HIS services and query processing strategies can be especially configured. An important topic is incorporated by the translation and execution of global queries in HIS federated system. To solve HIS related problems, distribution of subqueries and query execution semantics have to be reconsidered and readjusted as compared to ordinary distributed query execution.

Furthermore, promising solutions for middleware and software development have been explored. Adopting a service-based design approach for HIS architecture (based on CORBA/OMA) we may deal with an evolutionary development strategy for HIS and changes in middleware technologies. We have identified middleware requirements for HIS through services. Thus, we may insulate HIS architecture from middleware changes. In an implementation perspective, we have identified middleware solutions (e.g. CORBA, RMI, and dLIMIT) and their applicability to perform with HIS middleware services.

The SIDI project has developed a prototype for the exchange of patient information between the regional main emergency hospital of Porto Alegre (Hospital de Pronto Socorro) and the Bom Jesus regional community health centre [3]. This prototype is based on HL7 messages, addressing basically primary key queries on patients.

The considerations and results presented in this paper constitute the foundations for further work on SIDI, providing an alternative to the present prototype by offering high query capabilities, evolution of middleware technologies, CORBA/OMA architecture, and complex object and object identification mappings. Furthermore, valid paths that may be taken next comprise global data warehousing functionality as well as a migration path aiming at a tighter integration of HIS components to gradually achieve better performance and a better balancing of local vs. global services.

References

[1] S. Adali, K. Candan, Y. Papakonstantionu, V. Subrahmanian: Query Caching and Optimization in Distributed Mediator Systems. In *Proceedings of ACM SIGMOD*, Montreal, Canada, pages 137-148, 1996

[2] M. Amaral: Nomeclatura Médica e o Prontuário de Pacientes: Visão Estruturada, Organização Sistemática e Aplicações da Informática. In *Proceedings of Seminário sobre Prontuário Eletrônicodo Paciente*, São Paulo, Brazil, 1997 (in Portuguese)

[3] J. Castilho, J. Oliveira, C. Ribeiro: The SIDI Health System Project. In *Proceedings of Seminário de Avaliação do Projeto ProTeM Fase III*, Rio de Janeiro, Brazil, pages 407-420, 1999

[4] A. O. Freier, P. Karlton, P. C. Kocher: The SSL Protocol, Version 3.0 - Internet Draft. Netscape Communications, http://home.netscape.com/eng/ssl3/ssl-toc.html, 1996

[5] J. Gosling, B. Joy, G. Steele: The Java Language Specification. Addison-Wesley, New York, http:// java.sun.com/doc/language_specification.html, 1996

[6] G. Hamilton, R. Cattell: JDBC: A Java SQL API, Version 1.10, SUN Microsystems Corp., ftp://splash.javasoft.com/pub/jdbc-spec-0110.ps, 1996

[7] T. Härder, B. Mitschang, H. Schöning: Query Processing for Complex Objects. *Data & Knowledge Engineering* 7:181-200, 1992

[8] T. Härder, G. Sauter, J. Thomas: The Intrinsic Problems of Structural Heterogeneity and an Approach to their Solution. *The VLDB Journal* 8(1): 25-43, 1999

[9] Iona et al.: Patient Identification Service (PIDS). Initial Submission, Revision 2. OMG TC Document corbamed/97-06-01, June 1997

[10] R. E. Johnson: Reusing Object-Oriented Design. Technical Report UIUCDCS 91-1696, University of Illinois, 1991

[11] W. Kent, R. Ahmed, J. Albert, M. Ketachi, M.-C. Shan: Object Identification in Multidatabase Systems. In: Hsiao, D., Neuhold, E., Sacks-Davis, R. (eds.). *Interoperable Database Systems*, North-Holland, Amsterdam, 1993.

[12] W. Kim, J. Seo: Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer* 24(12): 12-18, 1991

[13] V. Kashyap, A. Sheth: Semantic and Schematic Similarities Between Database Objects: A Context- Based Approach. *The VLDB Journal* 5(4): 276-304, 1996

[14] H. Loeser, T. Härder: dLIMIT - A Middleware Framework for Loosely-Coupled Database Federations. In *Proceedings of 2nd Int. Conf. on World-wide Computing & Its Applications* (WWCA'98), LNCS 1368, Springer, Berlin, pages 412-427, 1998

[15] P. Missier, M. Rusinkiewicz: Extending a Multidatabase Manipulation Language to Resolve Schema and Data Conflicts. In *Proceedings of 6th IFIP TC-2 Working Conf. on Data Semantics*, Stone Mountain, Atlanta, Georgia, pages 93-115, 1995

[16] W. Rishel: HL7 Version 3: Overview. In *Proceedings of HL7 Plenary Meeting*, Washington, DC. August 1996. (http://www.mcis.duke.edu/standards/HL7/pubs/version3/Version3.htm)

[17] R. Rivest, A. Shamir, L. M. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* 21(2): 120-126, 1978

[18] J. Siegel: CORBA - Fundamentals and Programming. John Wiley, New York, 1996

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

[20] R. M. Soley: An Object Model for Integration. *Computer Standards & Interfaces* 15(2-3): 149-166, 1993