The VLDB Journal (1999) 8: 120–132

The VLDB Journal

Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution*

Goksel Aslan, Dennis McLeod

Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782, USA; e-mail: {gokselas,mcleod}@usc.edu

Edited by R. King · Received June 19, 1998 / Accepted April 20, 1999

Abstract. A key aspect of interoperation among data-intensive systems involves the mediation of metadata and ontologies across database boundaries. One way to achieve such mediation between a local database and a remote database is to fold remote metadata into the local metadata, thereby creating a common platform through which information sharing and exchange becomes possible. Schema implantation and semantic evolution, our approach to the metadata folding problem, is a partial database integration scheme in which remote and local (meta)data are integrated in a stepwise manner over time. We introduce metadata implantation and stepwise evolution techniques to interrelate database elements in different databases, and to resolve conflicts on the structure and semantics of database elements (classes, attributes, and individual instances). We employ a semantically rich canonical data model, and an incremental integration and semantic heterogeneity resolution scheme. In our approach, relationships between local and remote information units are determined whenever enough knowledge about their semantics is acquired. The metadata folding problem is solved by implanting remote database elements into the local database, a process that imports remote database elements into the local database environment, hypothesizes the relevance of local and remote classes, and customizes the organization of remote metadata. We have implemented a prototype system and demonstrated its use in an experimental neuroscience environment.

Key words: Federated databases – Semantic hetrogeneity resolution – Database interoperability – Database integration – Schema evolution

1 Introduction

A cooperative federated database system is a collection of autonomous and heterogeneous component database systems [14, 35], which unite into a loosely coupled form in order to interoperate. Interoperability between component database systems is achieved by means of the ability of individual components to actively and cooperatively share and exchange information units with other components in the federation.

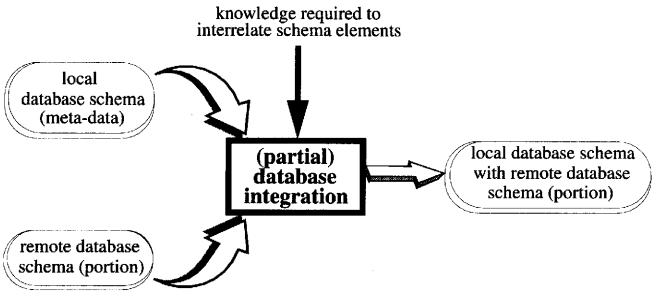

Fig. 1. The metadata folding problem

Information sharing and exchange necessitates data and metadata to be mediated across component databases in a federation. One way to achieve such mediation is to fold remote metadata (conceptual schema) into the local metadata, thereby creating a common platform through which information sharing and exchange becomes possible. This problem is termed as "the metadata folding problem" (Fig. 1); it can be stated more formally as: given two independently maintained databases, one local and one remote, it is the process of importing and customizing remote database elements (e.g., classes, attributes, and individual instances) into the local database in the presence of semantic differences between the two, so that remote information units can be accessed/manipulated from/within the local database environment. Importing means bringing remote database elements into, and making them accessible within the local database environment. Customization, on the other hand, refers to the process of reorganizing and/or tuning local and previously imported remote database elements by taking the real-world concepts they model and their corresponding contexts into account. In sum, the metadata folding problem is the problem of (partial) integration of remote and local databases in the presence of semantic conflicts.

* This research has been funded in part by the USC Integrated Media Systems Center (IMSC), a National Science Roundation Engineering Research Center, with additional support from the NIHM under grant no. 5P01MD/DA52194-02.

Previous approaches that are applicable to the metadata folding problem assume that either (1) knowledge required to relate interdatabase elements is available – the global schema approach, (2) derivable within the federation – approaches that employ semantic dictionaries or ontologies, (3) or obtainable from users – the multidatabase language approach. Most previous approaches propose a largely one-step solution, in which resolution of conflicts on the semantics of information units and integration of remote database elements into the local database are performed in a single pass. There are only a few attempts that consider the importance of the acquisition of knowledge required to interrelate information units in different databases. Assuming that knowledge required for semantic heterogeneity resolution and schema integration is always available or derivable results in either frequent user consultations on the semantics and interrelationships of information units and/or inconsistencies.

We propose here a uniform approach to the metadata folding problem, which recognizes the fact that such a knowledge may not be available. This results in a stepwise approach to semantic heterogeneity resolution: a stepwise (partial) integration scheme. In our approach, required knowledge is incrementally accumulated from information unit owners and maintainers (e.g., modelers, administrators, current users and application maintainers), who are the best experts on the semantics of information units they maintain in their databases. Interrelationships between database elements in different components are determined whenever sufficiently precise knowledge about their semantics is acquired.

The remainder of this paper is organized as follows. Section 2 examines previous work that is applicable to the metadata folding problem. Section 3 provides an example sharing scenario between a local and a remote database. It outlines key ideas on which our approach to the metadata folding problem is based, and specifies the specific sub-problems on which our approach focuses. Section 4 defines the canonical data model we employ as the common sharing and exchange language in the federation. Section 5 presents our approach and its sub-phases in detail. Section 6 describes implementation prototypes which realize our approach to the metadata folding problem. Finally, Sect. 7 presents a summary, conclusions, and potential enhancements to our work.

2 Related work

There are considerable research results that focus on the individual aspects of the metadata folding problem. A key focus has been on attempts to explicitly capture the semantics of information units within databases. For example, [11] describe what should be considered semantics and what should be considered structure in object-oriented databases. Semantics of information units are tightly coupled with the environment where information units reside. Therefore, information maintained in different database environments should be processed by respecting their corresponding environments, namely their contexts. [6, 33, 34, 36, 37] Semantic heterogeneity identification/resolution has been the focus of [12, 26], while [19] provides a comprehensive classification of

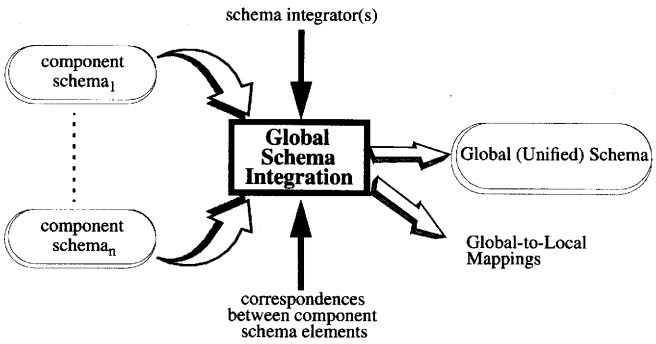

Fig. 2. Global schema approach

schematic conflicts and techniques to resolve them on a case-by-case basis.

There are three main categories of research that are directly applicable to the metadata folding problem. (1) Global schema approach: components agree on a common, federation-wide global schema before any information sharing and exchange takes place. Information unit semantics and interdatabase relationships are fixed on this schema. Information sharing and exchange occurs through this shared global schema. (2) Federated databases with semantic dictionary or ontologies: components agree on a pool of real-world concepts and relationships between concepts. Each component database is responsible for expressing the sharable portion of its conceptual schema in terms of this common vocabulary. Information sharing and exchange occurs by analyzing the actual concepts implied by individual database elements, by investigating interconcept relationships, and by deriving the meanings of unknown concepts when necessary. (3) Multidatabases with multidatabase languages: a multidatabase system employs a powerful language which is furnished with explicit primitives that enable a user to mediate through the component databases. Users in such an environment are assumed to be knowledgeable enough to be able to express their intentions by using this language statements in order to achieve information sharing and exchange.

Approaches assuming a global schema [1, 4, 7–10, 18, 20, 22, 31] (Fig. 2) address key issues such as generating a global schema physically or virtually (e.g., via a view mechanism), generating global schema-local schema mappings, usage of generalization primitive in schema integration, and the notions of equivalence of domains, classes, and attributes between databases. For instance, [7, 20] focus on the problem of constructing a global schema given correspondence assertions (conditions among classes, attributes, or composition hierarchies of object schemas in different components). Based on the correspondence assertions, integration rules are constructed, which use a set of primitive integration operators to achieve the integration.

The most serious limitation of global schema approach is that it requires too much and too broad global knowledge to generate the global schema. It also requires a large global structure to be maintained. The required integration effort is very high. Another limitation is the lack of flexibility in the relationships between component database elements, since all the possible relationships between schema elements in different components are fixed in the global schema. A

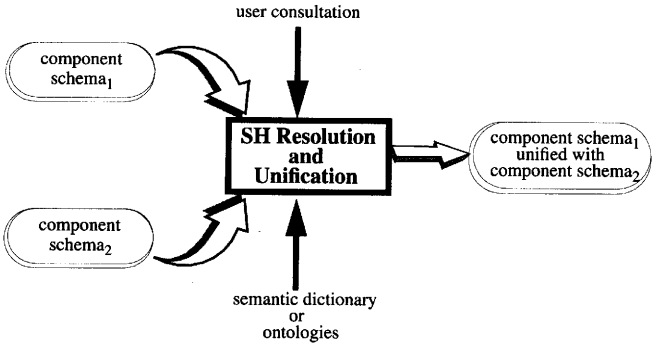

Fig. 3. Federated databases with global structures (semantic dictionary/ontologies)

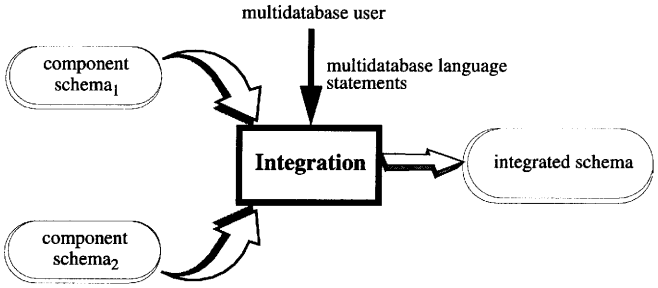

Fig. 4. Multidatabases with a multidatabase language

tightly coupled federated database with multiple federated schemas [35] is a generalization of the global schema approach, and it also suffers from the same limitations.

The federated databases with semantic dictionaries/ontologies approach [12, 13, 17, 26, 27, 40, 41] (Fig. 3) is based on the idea of agreeing on a collection of concepts and interconcept relationships federation-wide, and describing sharable portions of component databases in terms of this commonly understood set of concepts. When considered within the context of the metadata folding problem, this approach requires a moderate amount of global knowledge, global structures, integration effort, and actual exchange as compared with the global schema approach. The main limitation of this category is the difficulty of agreeing on a set of concepts and interconcept relationships in a federation environment that is dynamic (evolves).

In the third key approach to the metadata folding problem (Fig. 4), multidatabase systems offer multidatabase users with powerful multidatabase languages through which users can manipulate data in different non-integrated schemas[21, 23, 24, 30]. The MDSL [23] multidatabase language of MRDSM (a prototype multidatabase system) contains capabilities to join data in different databases, to broadcast user intentions over a number of database schemas, to flow data between databases, and to aggregate data from various databases. This approach, when applied to the metadata folding problem, requires a comparatively small amount of global structures to be maintained. Interrelationships of different databases are highly dynamic. Nevertheless, its requirement for database users to have and maintain a high degree of global knowledge about the remote information unit semantics constitutes a very important limitation; this likely results in high user effort during information sharing and exchange.

"Schema implantation and semantic evolution", our approach to the metadata folding problem, requires no global structures, although it does not rule out usage of such structures. Here, global knowledge required from a component in order to interoperate with other components is minimal, leading to a very low integration effort. Interrelationships between database elements in the federation are highly dynamic, and the effort that has to be spent during actual sharing and exchange is moderate.

Several other recent research results address partially the metadata folding problem with the desired characteristics defined above (e.g., minimum user involvement, minimum global structures, minimum global knowledge). For instance, the OBSERVER system [28, 29] addresses the problems of query processing and query reformulation in global information systems. In this study, information contents of components are captured in the form of metadata (e.g., semantic descriptions of heterogeneous repositories) in vocabularies (ontologies). The problem of different vocabularies is resolved by maintaining interrelationships between the terms in different ontologies. By contrast, our approach does not have any global structures which resemble ontologies and structures maintaining interontology relationships.

The SIMS project [2, 3] constructs a domain model, a hierarchical, terminological knowledge base, in order to describe application domain semantics. By contrast, we distribute the responsibility of maintaining information unit semantics into individual components, and assume an environment where there are multiple component databases which, unlike SIMS, may model overlapping or even independent application domains. The Carnot project at MCC [15, 39, 42–44, 47] bears similar limitations. A global schema or context in the form of a Cyc knowledge base is used as the federating mechanism. Here, a relationship between a domain concept from a local component and one or more concepts in the global context is expressed as an articulation axiom, a statement of equivalence between these contexts [15]. One of the services Carnot provides is the semantic services, whose purpose is to provide a global, enterprise-wide view of all the resources being integrated [43]. Another limitation is the difficulty of obtaining the declarative resource constraint base which contains interresource dependencies, consistency requirements, and consistency restoration strategies. The InfoSleuth project [5, 16, 45–48] investigates the use of Carnot technology in a more dynamically changing environment such as the Internet [5]. In such an ever-growing and ever-changing environment, information advertisement, information discovery, and collaboration between clients to fuse information from many information sources are necessary. InfoSleuth follows an agent-based approach to these problems, in which responsibility of carrying out different tasks are distributed over highly specialized agents. For example, InfoSleuth employs a special agent, called an "ontology server", which is responsible for managing the creation, update, and querying of multiple ontologies.

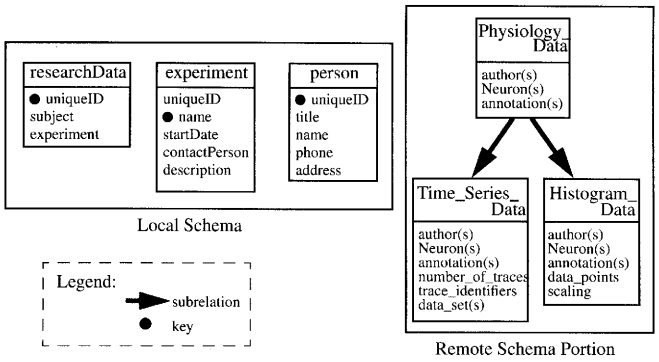

Fig. 5. Local and remote conceptual schemas

3 Schema implantation and semantic evolution

3.1 An example sharing scenario

In this section, we present an example sharing scenario between two federation components which maintain related information in their databases. Throughout the rest of this paper, we will illustrate individual phases of our mechanism, and new concepts we introduce by means of this example from the neuroscience domain.

Figure 5 shows an example of a local component which desires to expand its database with related remote information units. The local schema is managed by an object-relational DBMS. In the local database, information about experiments and people performing the experiments are recorded, as well as information about research data. Each experiment has a name, a start date, a description, and a contact person. Each person has a name, a title, a phone number, and an address. The researchData relation keeps related information on what experiment is performed on what subject. In addition, each relation has a unique identifier.

The remote component is interested in detailed analysis of result data associated with experiments. As a result, the Physiology_Data relation keeps record of different experimental data sets published in the literature. For each scientific data item, its authors, related neurons, and annotations are recorded. Moreover, two kinds of physiology data are identified (indicated by arrows as sub-relations in Fig. 5). In addition to the attributes of Physiology_Data, the Time_Series_Data relation records number of traces, a set of trace identifiers, and data sets, while data points and scaling information are kept in the Histogram_Data relation.

At some point in time, the local database may be more involved with experimental results, e.g., different kinds of data sets obtained from experiments. The need here would then be to extend the information content of the local database with the information content of the remote database. This allows the access and manipulation of remote information units within/from the local database. The key process then is to fold the remote conceptual schema into the local schema, which will take place within the local database context.

3.2 Schema implantation and semantic evolution approach

Schema implantation and semantic evolution employs metadata implantation and stepwise evolution techniques to interrelate database elements in different component databases, and to resolve conflicts on the semantics of database elements. It is a uniform approach to the problems of (partial) schema integration, semantic heterogeneity resolution, and schema customization for the purpose of ensuring database interoperability in federated databases. It is based on the following key ideas.

- The canonical data model employed in the federation and its constructs are important in making information unit semantics explicit.
- An incremental integration and semantic heterogeneity resolution process is utilized, wherein relationships between local and imported remote information units are determined whenever enough knowledge about their semantics is acquired, therefore, recognizing the possibility of incomplete, missing, and insufficient knowledge about information unit semantics in a federation.
- A partial integration scheme is used, in which semantics and context of local information units are dominant to that of remote information units, thus enabling multiple semantics to co-exist in a federation.

The metadata folding problem can be decomposed into five key sub-problems: (1) information discovery: remote database and the portion of the remote database that is of interest to local database needs should be located, (2) schema transformation: as component database systems may employ different metadata languages (data models) in order to model real-world entities, they need to be brought into a common formalism, a canonical data model, so that they can be compared, (3) semantic heterogeneity resolution: metadata, and data if necessary, should be analyzed in order to determine whether there exist relationships between schema elements in different components, their real-world counterparts (their meanings) should be well understood, and conflicts in their meanings should be resolved, (4) schema/instance importation: remote data and metadata should be brought into and made accessible to the local environment, (5) schema/instance customization: while respecting the semantics of imported and local information units, the organization of imported and local information units should be tuned so that imported information units fit into the local context. Our approach specifically covers subproblems (2) through (5), while the information discovery problem is addressed elsewhere [17, 27].

In our example in Fig. 5, local database user/administrator finds the relevant remote database and projects the remote database portion that is of interest to the local application. Both local and remote databases should be in a common language, the canonical data model of the federation. Therefore, they are translated from extended relational model to the canonical data model. After local and remote database schemas are analyzed for potential conflicts, remote schema portions (Physiology_Data, Histogram_Data, Time_Series_Data) and corresponding instances are imported into the local database. The last step is to establish concrete relationships between the local and imported remote schemas, e.g., where the Physiology_Data and its subtables belong within the local schema should be determined.

4 Heterogeneous Semantic Data Model (HSDM) as the canonical data model

We employ a semantically rich and expressive object-based data model, called HSDM (Heterogeneous Semantic Data Model), as the canonical data model in use federation-wide. HSDM supports traditional object-based notions such as object identity, classes, attributes, and a set of semantic primitives (classification, aggregation, and generalization). Moreover, it introduces new constructs in order to represent semantics within conceptual schemas. The purpose of HSDM is to ease information sharing and exchange among federation components by producing conceptual schemas where information unit semantics are easily understood. HSDM is specifically an extension of PDM (Personal Data Manager) [25].

An HSDM database is a collection of objects and relationships between objects. Each object has a name, a character string representation of the object, which functions as an object identifier. An HSDM class is a collection of similar objects with common attributes. An HSDM conceptual schema forms a generalization hierarchy representing sub-class–super-class relationships between classes, where class "Object" is the root. STRING, NUMBER, and their sub-classes (P_STRING, N_STRING, M_STRING, CHARACTER, INTEGER, REAL) constitute the predefined primitive classes, classes which do not have any user-defined attributes, in the class hierarchy. Distinguishing four subclasses of STRING (P_STRING, N_STRING, M_STRING, CHARACTER) helps resolution of certain conflicts between attributes. P_STRING (pure string) class can contain values which consist of a sequence of non-numeric characters. N_STRING (numerable string) represents values which consist of a sequence of numeric characters, an optional sign character ('+' or '−'), and an optional fixed point character ('.'). An M_STRING (mixed string) value is a sequence of numeric and non-numeric characters, provided that it has at least one numeric and one non-numeric character in it. Classes CHARACTER, INTEGER, and REAL correspond to usual data types present in most data models.

Three kinds of sub-class–super-class relationships are identified in HSDM. 1. Attribute defined: the sub-class has at least one attribute that is different than the ones inherited from its super-class and the ones defined by its siblings, 2. Predicate defined: the sub-class and its super-class have the same set of attributes; the values of one of the attributes are used to distinguish the sub-class instances from that of super-class and that of its siblings, 3. User defined: membership of sub-class instances cannot be determined without user involvement. We assume that in HSDM each class has at most one super-class.

Attributes in HSDM represent significant aspects of classes. They can be categorized as the ones that exist in original schemas before we apply our technique (application attributes), and others which are introduced during the application of our technique (complementary attributes). Unlike application attributes, complementary attributes may be associated with both classes and other attributes. Descriptor attributes are complementary attributes, which define an object class's (attribute's) meaning in natural language. Complementary attributes Kind, Unitofmeasure, Format, and Length are associated with primitive attributes (an attribute whose domain class is primitive) in order to clarify their meanings.

Every non-primitive object class in HSDM has a class key, and an instance key, which are both user-defined. The class key of an object class is a collection of attributes of that class whose existence conceptually distinguishes that class from others within the local context. Class keys allow determination of the most appropriate place a particular class belongs within the generalization hierarchy, and can be interpreted as conceptual class identifiers in this regard. Individual attributes appearing in a class key represent the most characteristic aspects of their classes, and help in finding an appropriate location for the class in the class hierarchy. Immediate sub-roots of the "object" class have single-attribute class keys: the attribute that is unique across all classes. The class key of a sub-class consists of class key of its super-class combined with either a unique attribute across all classes (in the case of attribute-defined sub-classes) or a predicate value that does not overlap with predicate values of all predicate-defined sub-classes defined on the same attribute (in the case of predicate-defined sub-classes). It is the existence of class keys and complementary attributes (Kind, Unitofmeasure, Format, and Length) which enables easy placement of remote classes into the local class hierarchy. An instance key is a collection of attributes whose value distinguishes an instance from others within a class.

HSDM contains a number of schema evolution primitives that provide capabilities such as renaming an attribute (class), dropping an attribute (class), adding a new attribute to a class, combining two classes, and making a class a super-class (sub-class) of another. These primitives are extensively used during restructuring the implanted remote conceptual schema during folding: once the relationship between a local and a remote class is determined, these primitives enable placement of remote class (instances) into their designated places in the local class hierarchy.

HSDM supports a number of null values corresponding to different kinds of uncertainties while modeling incomplete information. Initial null ($null_i$) corresponds to the initial unknown state of a newly introduced complementary attribute. The don't know null ($null_?$) corresponds to a missing attribute value. Finally, inapplicable null ($null_x$) signifies improper attachment of an attribute to a class.

5 Phases of schema implantation and semantic evolution

Figure 6 illustrates the schema implantation and semantic evolution approach to the metadata folding problem. Remote and local schemas, which are translated into HSDM, carry substantive knowledge that makes information unit semantics explicit. This is achieved by means of unique HSDM concepts like class keys, instance keys, and complementary attributes. After implanting remote database elements into the local database, knowledge required to interrelate them is acquired in a stepwise fashion during the semantic evolution phase. Acquired knowledge is used to obtain a new local database schema that is more tightly coupled with implanted remote schema elements than that of the previous version of the local database schema.

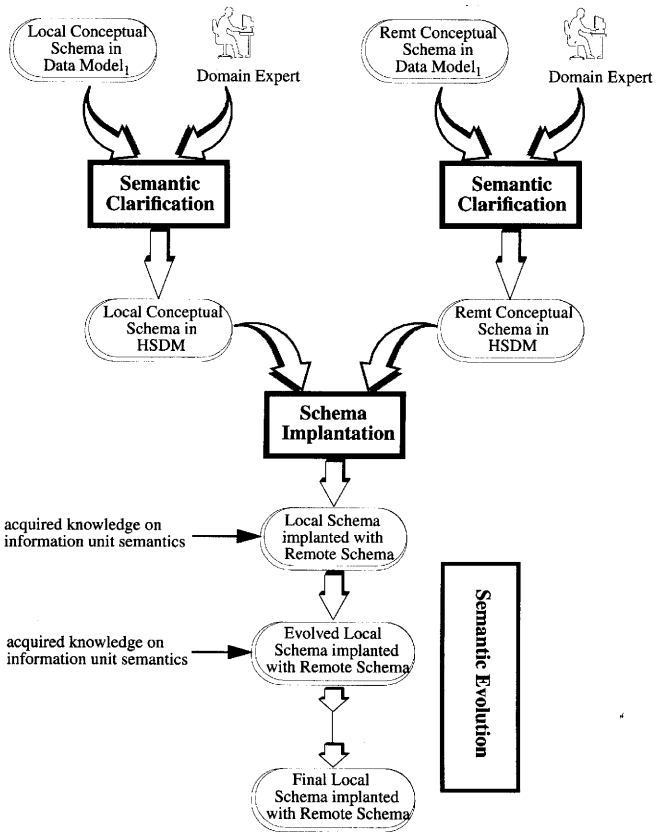

Fig. 6. Our approach: schema implantation and semantic evolution

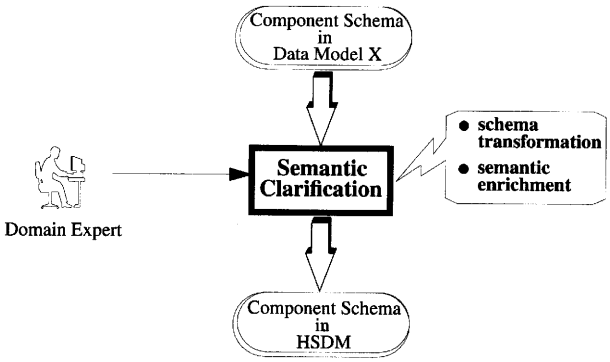

Fig. 7. Semantic clarification

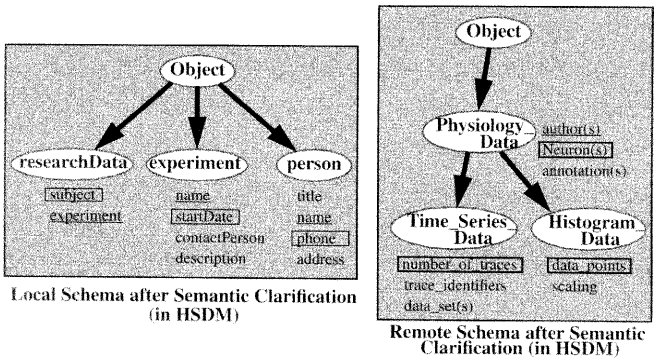

Fig. 8. Local and remote conceptual schemas after semantic clarification

The schema implantation and semantic evolution technique consists of three phases: (a) semantic clarification, (b) schema implantation, and (c) semantic evolution.

5.1 Semantic clarification phase

The inability of most data models to represent information unit semantics in an explicit, comparable, and easily interpretable manner necessitates explicit structures to maintain descriptions of and interrelationships between information unit semantics that may reside in different databases. In order to alleviate the need for such global structures, HSDM provides special constructs that augment the semantics of existing metadata (e.g., class keys, instance keys, descriptor attributes, the Kind attribute for primitive attributes).

The semantic clarification phase (Fig. 7) transforms a component conceptual schema from its native data model into HSDM (schema transformation), and augments it with additional information (semantic enrichment). Activities in this phase are performed for each component only once when the component enters into the federation, or when it decides to share and exchange information units with other components. In this process, descriptor attributes are created and attached to each class and each attribute. For primitive attributes, complementary attributes Kind, Unitofmeasure, Format, and Length are created in order to clarify their meaning. For each user-defined class in the class hierarchy, a class key and an instance key are determined with the help of a domain expert. In cases where there does not exist any potential attribute which can serve as the (part of the) class key or instance key, complementary attributes are introduced for this purpose. User-defined sub-classes are converted to attribute-defined or predicate-defined sub-classes, without loss of generality.

Figure 8 shows example local and remote schemas after the semantic clarification phase. Class keys of classes are shown as attributes enclosed by boxes, while instance keys are underlined. In the local schema, existence of attribute phone, the class key of person, conceptually distinguishes the person class from all other classes. The specification of instance keys indicates the application's point of view of the real world. For example, the local database distinguishes experiments by their name attribute values. Class keys and instance keys can contain complementary attributes, although such a need is not observed in our example.

5.2 Schema implantation phase

The schema implantation phase (Fig. 9) loosely integrates local and remote conceptual schema elements. It consists of two sub-phases: superimposing and hypothesis specification.

During superimposing (Fig. 10), remote schema elements are imported and super-imposed onto the local schema. Immediate sub-trees of the remote schema are added as new immediate sub-trees of class "Object" in the local schema. Predefined primitive classes constitute the directly sharable (agreed on) portion of conceptual schemas in the federation, and do not introduce any problems during the integration. User-defined primitive classes of local and remote databases, on the other hand, are interrelated by domain experts. Indi-

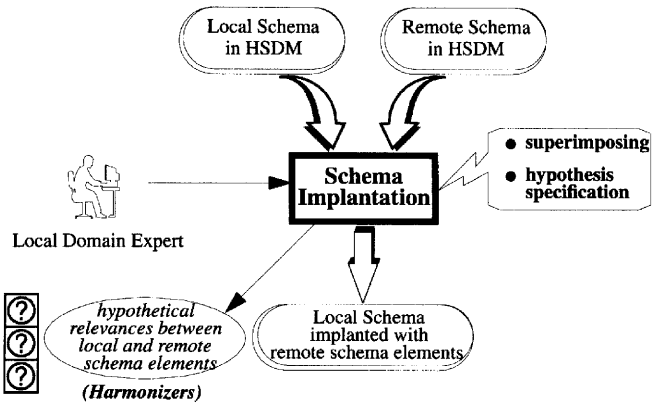

Fig. 9. Schema implantation

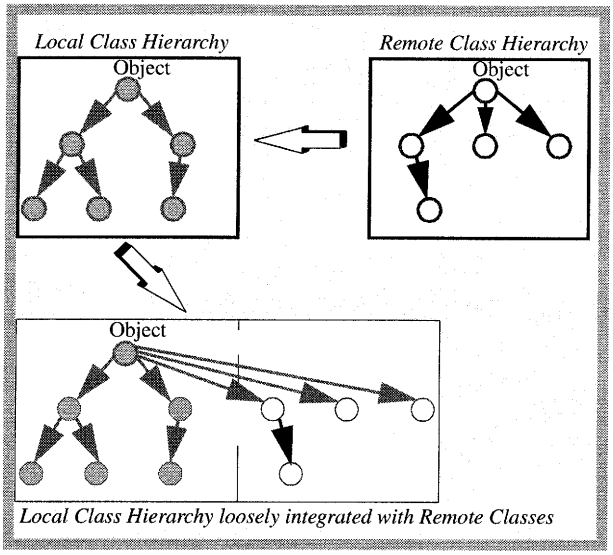

Fig. 10. Superimposing sub-phase

vidual remote instances are also imported and made available within the local database during this sub-phase.

The second sub-phase (Fig. 11) is termed hypothesis specification; here, a number of hypothetical relevances (harmonizers) are specified between local and remote classes by means of harmonizers. A harmonizer is a persistent structure that associates a local abstract class with a remote abstract class for the purpose of investigating whether or not they are semantically related (e.g., equivalence, sub-class, superclass, overlapping, or distinct). Each harmonizer has a name, an associated local class, an associated remote class, attribute equivalence assertions, known equivalences between the attributes of local and remote classes, and an object equivalence assertion, which specifies interobject relationships between local and remote class instances. The associated local class and associated remote class of a harmonizer are termed semantic peers.

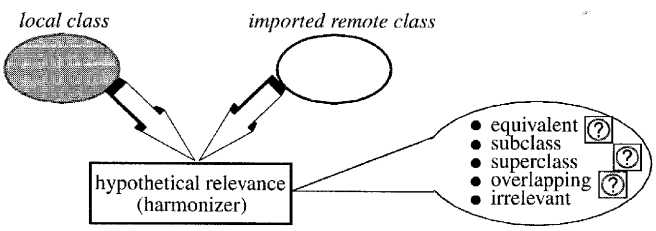

Fig. 11. Hypothesis specification sub-phase

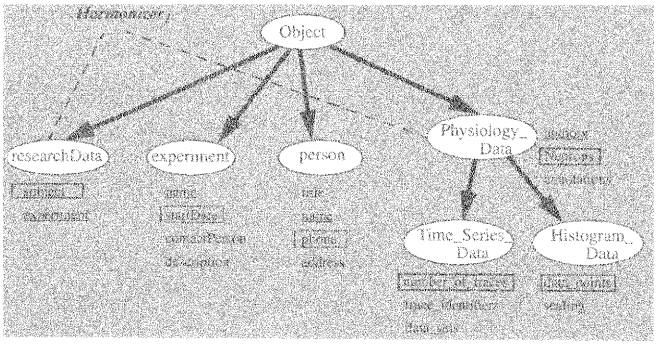

Fig. 12. Local schema implanted with remote schema portion

Harmonizer construction activity in the hypothesis specification phase is limited to the immediate sub-roots of the local conceptual schema implanted with the remote conceptual schema. Semantic peers are bound to each other in the form of a harmonizer for a common goal: to establish a semantic relationship between them. In order to realize this goal, we must establish that they can complement each other's structural, behavioral, and semantic aspects: with the construction of a harmonizer, the local class definition is expanded with attributes that exist in the remote class definition but not in the local class definition. Similarly, the remote class definition is expanded with attributes that exist in the local class definition but not in the remote class definition. Values for these newly introduced attributes are accumulated during the next phase.

With regard to our example, Fig. 12 shows the local schema implanted with the remote schema elements. (Attribute domains and details are not shown for simplicity.) A harmonizer, Harmonizer_1, is constructed between the local class researchData and the remote class Physiology_Data. The structure of Harmonizer_1 is shown in Fig. 13. Here, we see how complementary attributes are introduced by semantic peers on each other to complement each other's structural and semantic aspects. In Fig. 13, Harmonizer_1 is built as part of the schema implantation phase in order to investigate the relevance of the associated local class researchData and the associated remote class Physiology_Data. While no attribute equivalences are specified for this harmonizer_1, an object equivalence assertion depends on the instance key of the associated local class. According to this assertion, an instance of researchData class will be considered equivalent to another instance of Physiology_Data class if the value of attribute experiment of the local instance is equal to the value of attribute experiment of the remote instance.

After Harmonizer_1 is constructed, the local class researchData is added with complementary attributes, authors, neurons, and annotations, which originally belong to the remote class Physiology_Data. Accordingly, the remote class Physiology_Data is complemented with complementary attributes, subject and experiment, which originally belong to the local researchData.

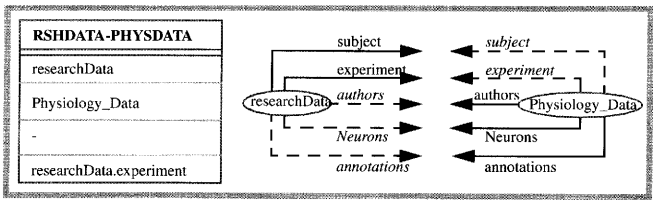

Fig. 13. The structure of Harmonizer$_1$

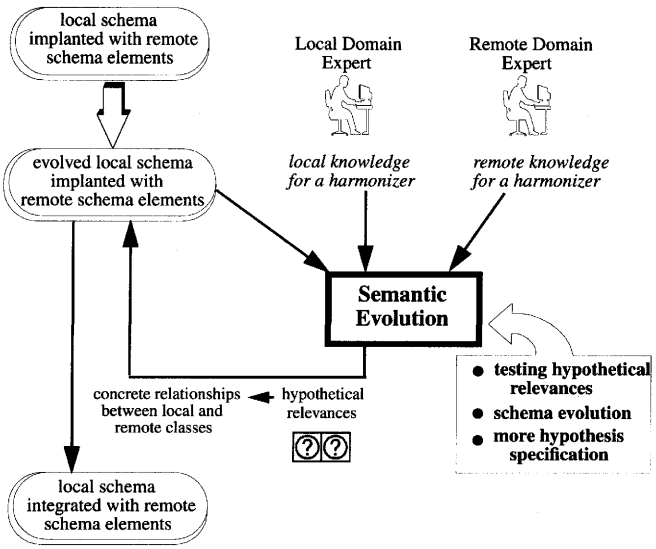

Fig. 14. Semantic evolution

According to the implicit contract between researchData and Physiology_Data as the consequence of building Harmonizer$_1$, (1) researchData class will supply meaningful values for the complementary attributes borrowed from the remote class Physiology_Data (authors, neurons, and annotations), and (2) Physiology_Data will supply meaningful values for complementary attributes borrowed from the local class researchData (subject and experiment).

5.3 Semantic evolution phase

The semantic evolution phase (Fig. 14) investigates whether previously hypothesized relevances hold, configures harmonizers for different local-class–remote-class combinations, and activates schema evolution primitives depending on the outcome of these investigations. With the construction of a harmonizer, the testing period starts for the harmonizer, during which information is accumulated for attributes imposed by local and remote classes on each other.

A sub-set of instances of a class that can be treated as representative for all instances is called a characteristic sub-set of that class. A characteristic sub-set of a class includes at least one instance from each of its sub-classes. During the testing period, local domain expertise tries to provide meaningful values for the attributes imposed by the remote class. Values are supplied for only the characteristic sub-set of the local class, not for all instances. Similarly, when a harmonizer is constructed, attributes that exist in the local class definition but not in the remote class definition are packed along with information that makes their semantics

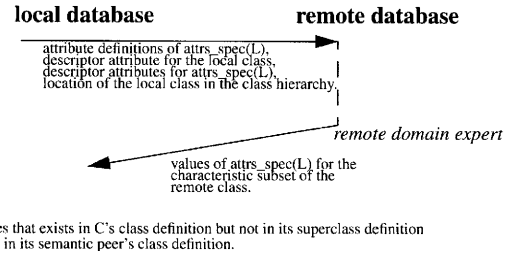

Fig. 15. Acquisition of knowledge about remote class instances

explicit (e.g., attribute definitions, descriptor attributes, location of the attribute's class within the class hierarchy), and are shipped to the remote database system (Fig. 15).

Based on the semantic information about attributes, the remote domain expert investigates if the remote class definition can be expanded with these new attributes by trying to provide meaningful attribute values for the characteristic sub-set of the remote class, and ships back these new values to the local database. New attribute values are chosen among permissible values in the domain classes of attributes or among different null values. For example, it is easy to choose values for a primitive attribute (an attribute whose domain class is primitive such as REAL, INTEGER, etc.), since primitive classes constitute the common part of each and every component database schema in the federation. However, an interclass attribute (an attribute between two user-defined classes) cannot always be given meaningful values from the domain class of the attribute. This is because the domain class of the attribute is defined in one context (local database) and may not be well known to the other context (remote database). The value of such an attribute can be chosen among different kinds of null values depending on whether or not the attribute makes sense for the specific instances in the characteristic sub-set.

Figure 16 shows the possible states of a harmonizer during the testing period. A harmonizer enters into the initial state when it is first created. The harmonizer sends local and remote requests to domain experts requesting additional knowledge about the associated local and associated remote classes of the harmonizer (complementary attribute values which were introduced by the semantic peers on each other). In the initial state, the harmonizer waits for this additional knowledge to arrive. Either the local knowledge or the remote knowledge may arrive first. If the local knowledge arrives first, the harmonizer jumps to the WR state (in Fig. 16), where it waits for the remote knowledge to arrive next. Similarly, the WL state is reached when the harmonizer is in the initial state, and it receives the remote knowledge. In both cases, the harmonizer state changes to RE. In the RE state, the harmonizer is ready to be evaluated, since it has already acquired both the local and the remote knowledge in order to determine the relevance of the semantic peers. Receiving the evaluate directive, it determines the relationship between semantic peers, and after a final user verification, it activates schema evolution primitives to establish the relationship on the conceptual schema. The state F is the final state, where this harmonizer is not needed any more, and can be deleted or archived.

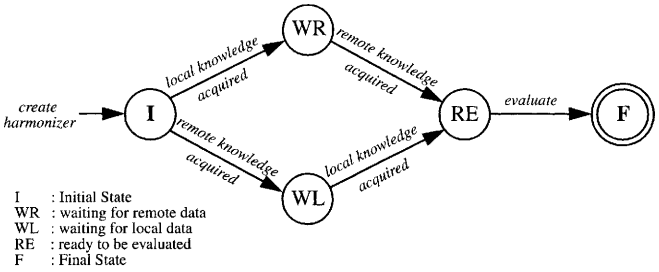

Fig. 16. State transition diagram for harmonizers

L: local class R: remote class

State	*meaningful local class key values for remote instances.*	Yes	Yes	Yes	Yes	No	No	No	No
	meaningful local non-class key values for remote instances.	Yes	Yes	No	No	Yes	Yes	No	No
	meaningful remote class specific attribute values for local instances.	Yes	No	Yes	No	Yes	No	Yes	No
Decision		EQUIVALENT(L,R)	SUPER-CLASS(L,R)	SUB-CLASS(L,R)	OVERLAPS(L,R)	SUB-CLASS(L,R)	DISTINCT(L,R)	SUB-CLASS(L,R)	DISTINCT(L,R)
Action		combine(L,R)	make-super(L,R)	make-sub(L,R)	make-has-common superclass(L,R)	make-sub(L,R)	build a new harmonizer	make-sub(L,R)	build a new harmonizer

Fig. 17. Possible states of a harmonizer and consequent actions after acquisition of necessary knowledge about semantics of local and remote classes

A harmonizer is said to reach the equilibrium state when it is able to suggest a relationship between semantic peers. Based on the collected attribute values, a decision is made regarding whether newly introduced remote (local) attributes make sense for the local (remote) class. For example, if the characteristic sub-set of the remote class has meaningful values for the class key attributes of the local object class, then it will be deduced that the remote class instances can satisfy the condition to be regarded as the instances of the local class. Figure 17 shows possible states, decisions made and consequent actions. As a result of this analysis, new harmonizers may be formed, existing ones may be propagated down into the class hierarchy, or individual schema evolution primitives may be activated to build explicit relationships between these semantic peers.

In our example, in order to test the relevance of the local and remote classes researchData and Physiology_Data we have built a harmonizer between these two classes. During the testing period of this harmonizer, the remote domain expert tries to supply values to subject and experiment attributes of the local class for the characteristic sub-set of the remote class. The local domain expert provides values to authors, neurons, and annotations attributes of the remote class for the characteristic sub-set of the local class. While the remote domain expert succeeds in providing meaningful values for the attributes subject and experiment, being unable to find meaningful remote class attribute values for some instances in the characteristic sub-set of the local class, the local domain expert provides inapplicable nulls for these attribute values. Therefore, evaluation of the harmonizer results in a super-class relationship between the local class and the remote class. The resulting conceptual schema is shown in Fig. 18.

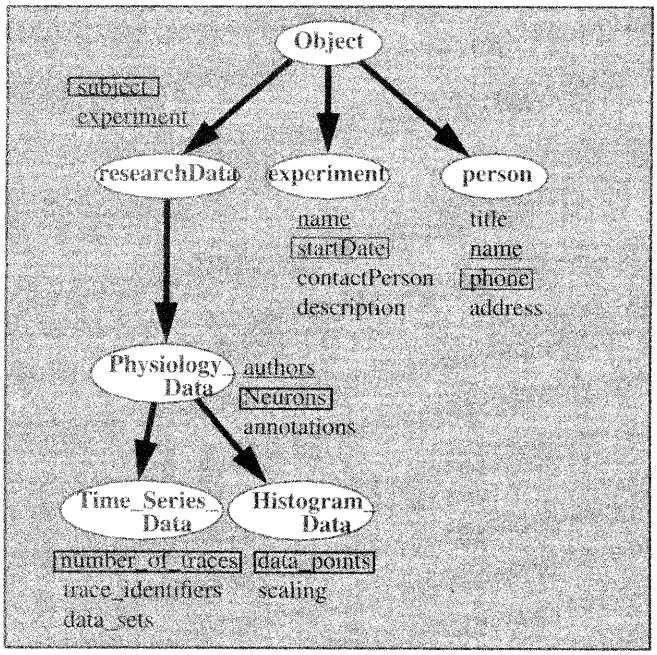

Fig. 18. Final local conceptual schema after semantic evolution

6 Implementation

Implementation of a full-fledged mechanism which realizes our approach requires two components: (1) translators that transform database schemas from various data models to HSDM, and (2) a database tool that manages the interactions between HSDM-managed databases. As there have been many efforts in the first area, we have focused on developing a tool which realizes the schema implantation and semantic evolution approach in a federation where all the component database systems employ HSDM as their data model. This prototype database tool is named the *HSDM Mediator*, and it was tested with examples from neuroscience domain within the context of USC Brain Project.

The prototype HSDM Mediator was built in two consecutive parts. As HSDM is an extension of PDM [25], first, we have implemented a database tool which allows modeling, representation, and management of information units using the PDM data model. Second, we have implemented the HSDM Mediator itself.

6.1 PDM implementation

We have implemented a simple, easy-to-use, object-based DBMS, called PDM, on top of ObjectStore [32]. PDM is based on a simple semantic data model proposed by McLeod and Lyngbaek [25]. It has a friendly user interface, in which users can browse the conceptual schema, kind[1] definitions, and individual instances. It also supports individual data manipulation operations such as insert, modify, and delete.

[1] A class in object-oriented data models is called an object kind in PDM

PDM supports a basic naming scheme, according to which every object in the database has a character string representation for identification purposes. This naming scheme can be seen as a primitive form of object identifier in object-based data models. Users can browse/create/manipulate PDM conceptual schemas and PDM databases by means of an important construct, namely Working Kind. A Working Kind is like a cursor in relational DBMSs. It can be bound to a number of instances of an object kind. Most of the operations work relative to the Working Kind.

PDM software was written in C++ with embedded ObjectStore [32] calls for database-related functionality, and embedded "curses" library function calls [38] for user-interface-related functionality. We have used ObjectStore as the storage sub-system in implementing PDM.

6.2 Implementation of the HSDM Mediator

The HSDM Mediator has two functions within a federation. First, it functions as a database modeling tool by enabling information units to be modeled/maintained in an HSDM database. Second, it allows HSDM databases to interoperate with each other under the principles of our approach. Like the PDM software, the HSDM Mediator software has been written in C++ with embedded ObjectStore statements for database-related functionality, and embedded "curses" library function calls for user-interface-related functionality. We have used ObjectStore as the storage sub-system in implementing the HSDM Mediator also.

In order to implement a prototype tool for HSDM, which also realizes our methodology, we have customized the PDM software. This has been achieved by implementing the capabilities HSDM supports but PDM lacks. In particular, we implemented the following capabilities of HSDM, and integrated it with the already existing PDM software. PDM does not offer any means to clarify schema semantics. HSDM, in contrast, allows specification of complementary attributes, class keys, and instance keys. Therefore, we have written code that enables classes to have instance and class keys, and that allows attachment of complementary attributes to classes and attributes. For explicit information unit semantics, we have implemented attribute-defined and predicate-defined sub-class mechanisms. Code required for interoperation purposes such as for importing remote classes and instances, and code for harmonizer construction and maintenance has been produced, and added into the software. Finally, schema evolution primitives we need to restructure conceptual schemas were added to the HSDM Mediator software.

Figure 19 shows the implementation architecture of our approach to the metadata folding problem, where the local component communicates with the remote component via the "Implant DB" operation and via requests for complementary information. The remote component provides remote data and metadata when "Implant DB" operation is invoked. It provides complementary attribute values for a particular harmonizer on request.

We now discuss the functionality of the HSDM Mediator prototype by describing individual operations it provides. We specifically focus on operations related to interoperability. The Mediator's functionality is divided into five categories corresponding to five sub-menus in the user interface: Database, Browse Schema, Working Kind, Manipulate DB, Interoperate (Fig. 20).

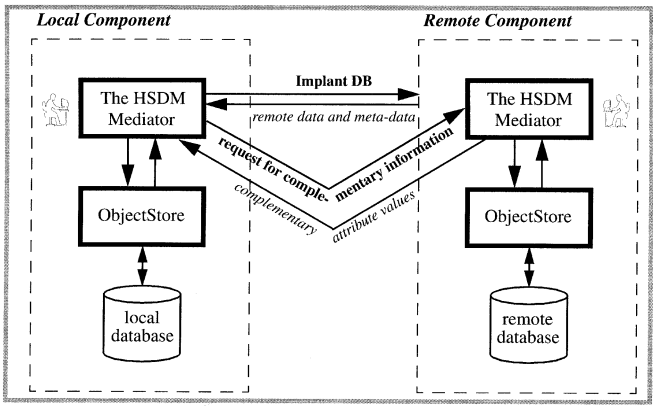

Fig. 19. Implementation architecture

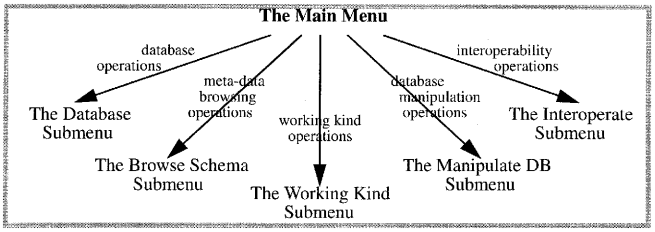

Fig. 20. Functionality of the HSDM mediator

In Fig. 20, the Database sub-menu contains menu items which operate on databases, such as create, open, and status. The Browse Schema sub-menu enables users to browse the class hierarchy, as well as to browse class definitions. Indentation, highlighting, and underlining techniques are used to display the class hierarchy on the screen. The third sub-menu in the user interface of the HSDM Mediator is the Working Kind sub-menu. This sub-menu enables manipulation and retrieval of the Working Kind. The Manipulate DB sub-menu includes operations that change the database state in a permanent manner such as creating a kind, creating an instance, or modifying an instance.

The Interoperate sub-menu (Fig. 21) enables HSDM-managed components to interoperate. The Implant DB is used to implant remote databases into a local database. The operation implants a remote database into the local database

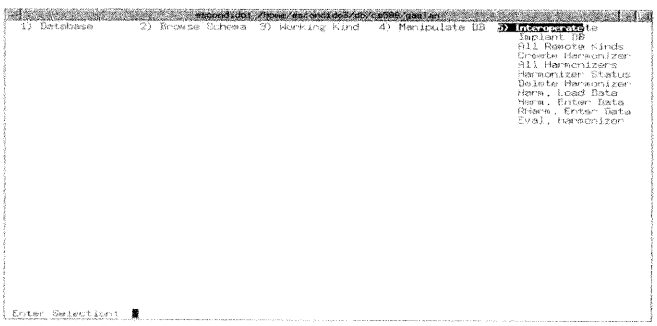

Fig. 21. The interoperate sub-menu

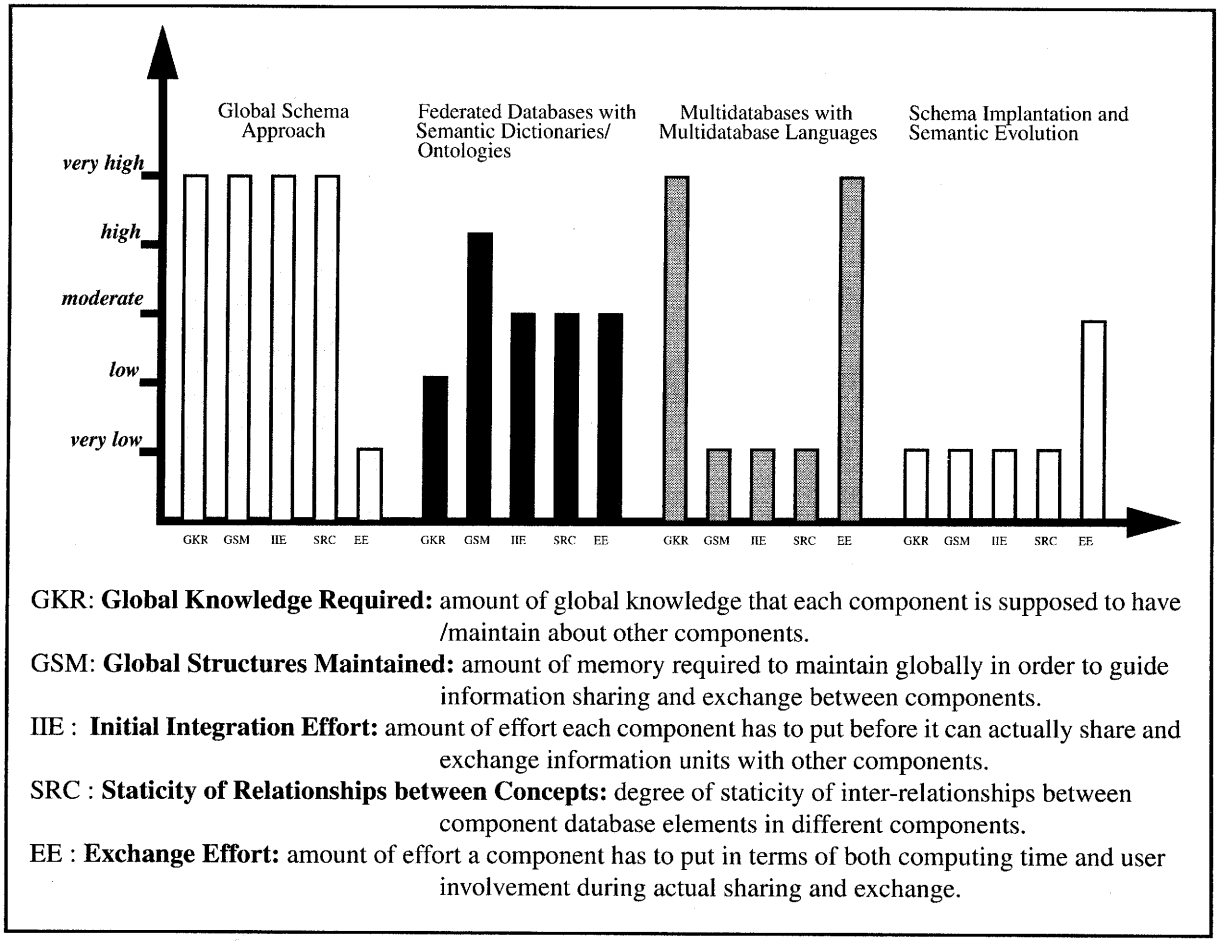

Fig. 22. Possible solutions to the metadata folding problem

by importing the class hierarchy, kinds, and instances of the specified remote database. The All Remote Kinds operation is necessary for displaying imported remote kinds that were not tied with the local class hierarchy yet. The databases from which these remote kinds originate, and any harmonizers in which they are involved are also displayed.

The remaining operations are for harmonizer processing. Using the Create Harmonizer operation, users specify hypothetical relevances, harmonizers, between local and imported remote kinds. All Harmonizers displays all the harmonizers under investigation. Harmonizer Status, on the other hand, displays information about a specific harmonizer such as its name, associated local kind, associated remote kind, and its status (waiting for local and remote data, waiting for local data, waiting for remote data, ready to be evaluated, and evaluated). Harmonizers are persistent structures in our prototype. They should be disposed of or archived when they complete performing their functions (e.g., after they are evaluated); Delete Harmonizer is used for this purpose.

The Harm. Enter Data operation enables the local domain expert to enter complementary attribute values for the characteristic sub-set of the local kind. Using the RHarm. Enter Data operation, the remote domain expert provides complementary attribute values for the characteristic sub-set of the remote kind. Supplied values are sent back to the local component. When a harmonizer is in the "ready to be evaluated" state, the Eval. Harmonizer operation is activated. As a result, accumulated information is analyzed, and a suggestion is made regarding the possible relationship between the associated local kind and the associated remote kind of this harmonizer. After the user is prompted with the consequences of the suggested action, he/she is expected to confirm this suggestion. When the user confirms a suggested action, individual schema evolution primitives are activated to establish the suggested relationship between the associated local and remote kinds of the harmonizer.

7 Discussion, conclusions and research directions

In this paper, we have described an approach to the metadata folding problem, which emphasizes incremental acquisition of knowledge required to fold a remote conceptual schema onto a local one for the sake of information sharing and exchange. One observation we have made is the difficulty of maintaining and/or agreeing on global structures in a federated database system environment. Consequently, we do not assume any global entities in our approach, except for HSDM.

We employ hypothetical processing (hypothesizing that two classes are related via equivalence, specialization, generalization, overlapping, or irrelevance, and endeavor to prove that such a hypothesis holds for a small, but very typical

sub-set of instances) while investigating whether local and remote classes have the ability to complement each other's semantic and structural aspects. This is in order to form a relationship between them in the local class hierarchy, therefore incrementally placing remote conceptual schema elements into the local class hierarchy where they make sense.

When considered within the context of the metadata folding problem, each approach to the metadata folding problem has its pros and cons, which are summarized in Fig. 22. For example, the global schema approach requires too much global knowledge from federation users both during integration and during actual sharing and exchange. It requires huge amounts of space to store the global schema, and the initial integration effort that has to be spent is very costly and prohibitive. Furthermore, semantics and interrelationships of schema elements are not dynamic because of the fixed global schema. Nevertheless, exchange effort is very low, since every possible sharing pattern is fixed and obvious in the form of the global schema. The schema implantation and semantic evolution approach on the other hand requires minimal global knowledge, minimal global structures, and minimal initial integration effort. Interrelationships of schema elements in different components are highly dynamic since they are not tightly bounded. However, it necessitates moderate exchange effort to be spent because of the need to acquire additional attribute values for classes that are hypothesized to be related.

The ultimate success of our approach, and of any approach claiming to provide a solution to the metadata folding problem, depends on the amount of user interaction/consultation required. Acquiring additional information for only a characteristic sub-set of a class in our approach is a direct result of this concern. One cause for still too much user interaction in our scheme is choosing harmonizers in a way that results in irrelevance. Currently, our approach depends on user intuition in building initial harmonizers. It would be worthwhile to consider building a mechanism that analyzes data and metadata of remote and local databases, and suggests potential harmonizers to users. This would greatly reduce required user input, since it will reduce the number of harmonizers to be built in order to integrate the schemas. Another shortcoming of our approach is that we depend on structural properties of a class definition while investigating its relevance to a remote class. An extension which considers behavioral properties (methods) of class definitions as well would contribute additional power. Still another interesting extension would be to study implications of multiple inheritance within this framework.

References

1. Abiteboul S, Bonner A (1991) Objects and views. In: Clifford J, King R (eds) Proceedings of ACM SIGMOD, 1991, Rec 20(2): 238–247
2. Arens Y, Knoblock CA, Hsu C (1996) Query processing in the SIMS Information Mediator. In: Tate A (ed) Advanced Planning Technology. AAAI Press, Menlo Park, Calif.
3. Arens Y, Knoblock CA, Shen W (1996) Query Reformulation for Dynamic Information Integration. J Intelligent Inf Syst 6(2/3): 99–130
4. Batini C, Lenzerini M, Navathe S (1986) A comparative analysis of methodologies for database schema integration. ACM Comput Surv 18(4): 323–364
5. Bayardo R, Bohrer W, Brice R, Cichocki A, Flowler G, Helal A, Kashyap V, Ksiezyk T, Martin G, Nodine M, Rashid M, Rusinkiewicz M, Shea R, Unnikrishnan C, Unruh A, Woelk D (1997) InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments. In: Peckham J (ed) Proceedings of ACM SIGMOD International Conference on Management of Data, 1997, Tucson, Arizona. 26(2): 195–206
6. Bresson S, Goh CH, Fynn K, Jakobisiak M, Hussein K, Kon HB, Lee T, Madnick SE, Pena T, Qu J, Shum AW, Siegel M (1997) The Context Interchange Mediator Prototype. ACM SIGMOD Rec 26: 525–527
7. Chen ALP, Koh JL, Kuo TCT, Liu CC (1995) Schema integration and query processing for multiple object databases. Integrated Comput Aided Eng (Special Issue on Multidatabase and Interoperable Systems) 2(1): 21–34
8. Czejdo B, Rusinkiewicz M, Embley D (1987) An approach to schema integration and query formulation in federated database systems. In: Proceedings of the 3rd IEEE Conference on Data Engineering, 1987, Los Angeles, CA. IEEE Comp Society, pp 477–484
9. Dayal U, Hwang H (1984) View definition and generalization for database integration in a multidatabase system. IEEE Trans Software Eng 10(6): 628–644
10. Elmasri R, Navathe S (1984) Object integration in logical database design. In: Proceedings of IEEE Computer Society 1st International Conference on Data Engineering, 1984, Los Angeles, CA. IEEE Comp Society, pp 426–433
11. Geller J, Perl Y, Neuhold EJ (1991) Structure and semantics in object-oriented database class specifications. ACM SIGMOD Rec 20(4): 40–43
12. Hammer J, McLeod D (1993) An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. Int J Intelligent Coop Inf Syst 2(1): 51–83
13. Hammer J, McLeod D, Si A (1994) Object discovery and unification in a federated database system. Technical Report USC-CS. Computer Science Department, University of Southern California, Los Angeles, Calif.
14. Heimbigner D, McLeod D (1985) A federated architecture for information management. ACM Trans Off Inf Syst 3(3): 253–278
15. Huhns M, Jacobs N, Ksiezyk T, Shen W, Singh M, Cannata P (1992) Enterprise Information Modeling and Model Integration in Carnot. In: Petrie CJ jr (ed) Enterprise Integration Modeling: Proceedings of the First International Conference, 1992. MIT Press, Cambridge, Mass.
16. Jacobs N, Shea R (1996) The Role of Java in InfoSleuth: Agent-based Exploitation of Heterogeneous Information Resources. In: IntraNet96 Java Developers Conference, April 1996
17. Kahng J, McLeod D (1996) Dynamic classificational ontologies for discovery in cooperative federated databases. In: Proceedings of the 1st International Conference on Cooperative Information Systems, June 1996, Brussels, Belgium. IEEE-CS Press
18. Kaul M, Drosten K, Neuhold EJ (1990) Viewsystem: Integrating heterogeneous information bases by object-oriented views. In: Proceedings of International Conference on Data Engineering 6, 1990, Los Angeles, CA. IEEE Comp Society, pp 2–10
19. Kim W, Choi I, Gala S, Scheevel M (1993) On resolving schematic heterogeneity in multidatabase systems. Distrib Parallel Databases 1(3): 251–279
20. Koh JL, Chen ALP (1993) Integration of heterogeneous object schemas. In: Elmasri R, Kouramojian V, Thalheim B (eds) Proceedings of the 12th International Conference on Entity Relationship Approach, 1993, Lecture notes in CS, Vol 823, Springer, pp 297–314
21. Krishnamurthy R, Litwin W, Kent W (1991) Language features for IEEE interoperability of databases with schematic discrepancies. In: Clifford J, King R (eds) Proceedings of ACM SIGMOD International Conference on Management of Data, 1991, Denver Colo. SIGMOD Record 20(2): 40–49
22. Larson J, Navathe SB, Elmasri R (1989) A theory of attribute equivalence in databases with application to schema integration. IEEE Trans Software Eng 15(4): 449–463
23. Litwin W, Abdellatif A (1986) Multidatabase interoperability. IEEE Comput 19(12): 10–18
24. Litwin W, Mark L, Roussopoulos N (1990) Interoperability of multiple autonomous databases. ACM Comput Surv 22(3): 267–293

25. Lyngbaek P, McLeod D (1984) A personal data manager. In: Dayal U, Schlageter G, Seng LH (eds) Proceedings of the International Conference on Very Large Data Bases, 1984, Singapore Morgan Kaufmann, pp 14–25
26. McLeod D (1991) The identification and resolution of semantic heterogeneity in multidatabase systems. In: International Workshop on Interoperability in Multidatabase Systems, 1991, Kyoto, Japan
27. McLeod D, Si A (1995) The design and experimental evaluation of an information discovery mechanism for networks of autonomous database systems. In: Yu PS, Chen ALP (eds) Proceedings of IEEE International Conference on Data Engineering, 1995, Taipei, Taiwan. IEEE Comp Society, pp 15–24
28. Mena E, Kashyap V, Illarramendi A, Sheth A (1996) Managing Multiple Information Sources through Ontologies: Relationship between Vocabulary Heterogeneity and Loss of Information. In: Baader F, Buchheit M, Jeusfeld MA, Nutt W (eds) Proceedings of the Third Workshop Knowledge Representation Meets Databases (KRDB), 1996, Budapest, Hungary. CEUR Workshop Proceedings No. 4
29. Mena E, Kashyap V, Sheth A, Illarramendi A (1996) OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In: Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS), 1996, Brussels, Belgium. IEEE-CS Press, pp 14–25
30. Missier R, Rusinkiewicz M (1995) Extending a Multidatabase Manipulation Language to Resolve Schema and Data Conflicts. In: Meersman R, Mark L (eds) Proceedings of the Sixth IFIP TC-2 Working Conference on Database Semantics (DS)-6, 1995, Stone Mountain, Atlanta, Georgia, USA, pp 93–115
31. Motro A (1987) Superviews: Virtual integration of multiple databases. IEEE Trans Software Eng 13(7):785–798
32. Object Design Inc (1993) ObjectStore User Guide: Library Interface, Release 3.0. Object Design Inc.
33. Sciore E, Siegel M, Rosenthal A (1992) Context Interchange using meta-attributes. In: Finin TW, Nichola CK, Yesha Y (eds) First International Conference on Information and Knowledge Management, 1992, Lecture notes in CS, Vol 752; Springer, Baltimore, Md, pp 377–386
34. Sciore E, Siegel M, Rosenthal A (1990) Using semantic salues to facilitate interoperability among heterogeneous information systems. ACM Comput Surv 22(3): 183–236
35. Sheth A, Larson J (1990) Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput Surv 22(3): 183–236
36. Siegel M, Madnick SE (1991) A metadata approach to resolving semantic conflicts. In: Lohman GM, Sernadas A, Camps R (eds) Proceedings of the International Conference on Very Large Data Bases, 1991, Barcelona, Catalonia, Spain. Morgan Kaufman, pp 133–145
37. Siegel M, Madnick SE (1991) Context Interchange: sharing the meaning of data. ACM SIGMOD Rec 20(4): 77–78
38. Strong J (1986) Programming with curses. O'Reilly & Associates, Sebastopol, Calif.
39. Tomlinson C, Lavender G, Meredith G, Woelk D, Cannata P (1992) The Carnot Extensible Services Switch (ESS) -Support for Service Execution. In: Petrie CJ jr (ed) Enterprise Integration Modeling: Proceedings of the First International Conference, 1992. MIT Press, Cambridge, Mass.
40. Tsai PSM, Chen ALP (1994) Concept hierarchies for database integration in a multidatabase system. In: 6th International Conference on Management of Data, 1994, Bangalore, India
41. Wiederhold G (1994) Interoperation, mediation and ontologies. In: Workshop on Heterogeneous Knowledge-Bases, 1994. W3, pp 33–48
42. Woelk D, Shen W, Huhns M, Cannata P (1992) Model-Driven Enterprise Information Management in Carnot. In: Petrie CJ jr (ed) Enterprise Integration Modeling: Proceedings of the First International Conference, 1992. MIT Press, Cambridge, Mass.
43. Woelk D, Cannata R, Huhns M, Shen W, Tomlinson C (1993) Using Carnot for Enterprise Information Integration. In: Second International Conference on Parallel and Distributed Information Systems, January 1993, San Diego, Calif. IEEE-CS, pp 133–136
44. Woelk D (1994) Carnot Intelligent Agents and Digital Libraries. In: Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries, June 1994
45. Woelk D, Tomlinson C (1994) The InfoSleuth Project: Intelligent Search Management via Semantic Agents. In: Second International World Wide Web Conference, October 1994, Chicago, USA
46. Woelk D, Tomlinson C (1995) InfoSleuth: Networked Exploitation of Information Using Semantic Agents. In: COMPCON Conference, March 1995, San Francisco, Calif. IEEE-CS, pp 147–152
47. Woelk D, Tomlinson C (1995) Carnot and InfoSleuth: Database Technology and the World Wide Web. In: Carey MJ, Schneider DA (eds) ACM SIGMOD Int. Conference on the Management of Data, May 1995, San Jose, Calif., SIGMOD Record 24(2):443–444
48. Woelk D, Huhns M, Tomlinson C (1995) InfoSleuth Agents: The Next Generation of Active Objects. MCC Technical Report INSL-054-95. MCC, Austin, Tex.