Object Discovery and Unification in Federated Database Systems*

Joachim Hammer, Dennis McLeod and Antonio Si

Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, USA
{joachim,mcleod,asi}@cs.usc.edu

Abstract

A key challenge in sharing-oriented information management environments, such as networks of heterogeneous, autonomous database systems, is to provide capabilities to allow information units and resources to be flexibly and dynamically combined and interconnected, while at the same time preserving the investment in and the autonomy of each individual component. The research described here specifically focuses on two key aspects of this: (1) how to discover the location and content of relevant, nonlocal information units, and (2) how to identify and resolve the semantic heterogeneity that exists between related information in different database components. Our approach serves as a basis for the sharing of related concepts through partial meta-data (conceptual schema) unification without the need for a global view of data. We demonstrate and evaluate our approach using the Remote-Exchange experimental prototype system [5], which supports information sharing and exchange from the above perspective.

1 Introduction

Support for interoperability among autonomous, heterogeneous data/knowledge base systems is emerging as a key information management problem for the 1990s. Cooperative work, computer-based manufacturing, scientific databases, and traditional data processing are only a few of the environments where collaboration among the individual systems is desired [17, 19, 24]. Proposed architectures to address the interoperability problem range from tightly coupled composite approaches in which individual databases are integrated into a centralized global database [1], to loosely coupled federated environments wherein

*This research was supported in part by NSF grant IRI-9021028.

information is shared among individual database systems while retaining their autonomy [11]. In this paper, we adopt the general context of a collection of cooperating, heterogeneous, autonomous database systems (DBSs); this is termed a *federated database system* (FDBS) or *federation* for short. Each individual database system constituting a federation is henceforth termed a *component database system* (or *component* for short). Within this framework, we will demonstrate a new, modular facility for discovering the location and content of related, nonlocal (remote) information, and for identifying and resolving the semantic heterogeneity that exists between related information in the FDBS.

The remainder of this paper is organized as follows. In Section 2, we review related work. In Section 3, we introduce an example of a real-world sharing scenario using a federation of collaborating scientists conducting research on macromolecules. In Section 4, we describe how we achieve interoperability among individual components using the collaborating scientists example. We introduce important terms and concepts used in this paper and provide the object context in which we have couched our research. Sections 4.1 through 4.4 describe in greater detail how our approach to information sharing functions, focusing on the individual modules of the mechanism. Finally, Section 5 contains concluding observations with a critical evaluation of our results and their potential impact.

2 Related Research

The term "heterogeneous databases" was originally used to distinguish work that included database model and conceptual schema heterogeneity from work on "distributed databases[1]" which addressed issues solely related to distribution [2]. Recently, there has been a resurgence in research in the area of heterogeneous database systems (HDBSs). Work in this area can be characterized by the different levels of integration of the component DBSs and by different levels of global (federation) services. In Mermaid [23], for example, which is considered a tightly coupled HDBS, component database schemas are integrated into one centralized global schema with the option of defining different user views on the unified schema. While this approach supports pre-existing component databases, it falls short in terms of flexible sharing patterns. Furthermore, the integration process is expensive and complicated, and tends to be difficult to change.

The federated architecture proposed in [11], which is similar to the multidatabase architecture of [16], involves a loosely coupled collection of database systems, stressing autonomy and flexible sharing patterns through inter-component negotiation. Rather than using a single, static global schema, the loosely coupled architecture allows multiple import schemas, enabling data retrieval directly from the exporter and not indirectly through some central node as in the tightly coupled approach.

One common approach to interoperability is to reason about the meaning and resemblance of heterogeneous objects in terms of their structural representation. In Larson et

[1]The term distributed database is used here as it has been mainly used in the literature denoting a relatively tightly coupled, homogeneous system of logically centralized, but physically distributed component databases.

al. [15], the meaning of an attribute is approximated in terms of its value type (set of possible values), cardinality constraints, integrity constraints, and allowable operations. However, one can argue that any such set of characteristics does not sufficiently describe the real-world meaning of an object, and thus their comparison can lead to unintended correspondences or fail to detect important ones. Other promising methodologies that have been developed include heuristics to determine the similarity of objects based on the percentage of occurrences of common attributes [10, 22]. More precise techniques use classification for choosing a possible relationship between classes [21].

In addition to methods primarily utilizing schema knowledge, techniques based upon the use of semantic knowledge (based on real-world experience) have also been investigated [6, 12]. These approaches typically assume the existence of a real world knowledge base which serves as a global schema to which every local schema is mapped. The similarity between different objects is determined by the use of techniques such as spanning trees in the knowledge base. The approach adopted in [6] associates a fuzzy value with each relationship in the knowledge base indicating the degree of fuzziness of that relation with respect to the concept/type to which the relation belongs; the set of fuzzy values also acts as a basis for quantitative similarity measurement between two objects. These approaches, however, lack the ability to tailor the identification process to the context of user request. Further, this approach assumes the existence of the centralized knowledge base. We believe that a useful approach is for the centralized knowledge base to only contain information actively used to support sharing within the federation and thus, as illustrated in our mechanism, should be dynamically built tailored to the federation.

A different approach proposed by Kent [14] uses an object-oriented database programming language to express mappings among different similar concepts that allow a user to view them in some integrated way. It of course remains to be seen if a language that is sophisticated enough to meet all of the requirements given can be developed in the near future.

A very recent approach to interoperability by Mehta et al. [18] uses so-called path-methods to explicitly create inter-component and inter-object mappings between source and target object classes in order to retrieve and update related data objects. The obvious drawback of this approach is the large overhead in calculating and maintaining the mappings, which may be impractical for large federations with extensive and/or dynamic sharing patterns. This approach of course also requires the determination of the relationship between objects belonging to different components.

3 A Federation of Collaborating Scientists

Consider the following scenario involving a group (federation) of collaborating scientists, each maintaining a database that contains information about macromolecules. Figure 1 shows a snapshot of the information stored in each database. Since each database is managed by a different scientist (or group thereof), the contents of these databases reflect the different foci and interests of their creators/owners. We can see, for example, that

Figure 1: A Federation of Collaborating Scientists

component D is the only one with information on both genetic and protein sequences. All other components maintain either genetic or protein information with various levels of detail.

It would clearly be beneficial to a scientist if information in his/her database could be dynamically shared and exchanged; for instance, a scientist could efficiently reuse protein data that had already been sequenced (decoded) by other components for his/her experimental work. A difficulty in finding a solution to information sharing in a federation of this kind stems from the conflicting nature of sharing and autonomy. On one hand, each scientist would like to share information with others; on the other hand, the same scientist would still like to retain autonomy over his/her own database with respect to organization and information release, e.g., control the information it is willing to "export" to other components. This capability of exporting only a specific portion of a component's database is particularly important in our federation of collaborating scientists where each component would probably not be willing to release any information that has not yet been published or fully validated. For simplicity here, we assume that all the information stored in a specially "marked" section of a component's database ("export schema") is available to every other component in the federation.

The level of interoperability that can be achieved within such a federation depends largely upon two key capabilities:

1. the ability of a component to identify and locate potentially appropriate nonlocal information with respect to its needs (the *discovery problem*); and

2. as requested remote information is identified and located, the ability to fold it into the local system framework (the *unification problem*).

Figure 2: The Remote-Exchange sharing architecture

The focus of this work is to address the above two problems. Several other important issues such as security (access control) and the update of shared data are not directly addressed in this paper.

4 Achieving Interoperability in Remote-Exchange

In order to provide a context for our approach to discovery and (local) unification, Figure 2 illustrates the architecture of the Remote-Exchange experimental system. For sharing and exchange of information to take place among the components, some common model for describing the sharable data must be utilized. To this end, we employ a **M**inimal **O**bject **D**atabase **M**odel (MODM), which contains the essential basic features found in existing object-based and semantic database systems. Briefly, MODM is a functional object-based model supporting complex objects, type membership, subtype to supertype relationships, inheritance of functions (methods) from supertype to subtype, and user-definable functions. An advantage of using an object-based common database model is the possibility of sharing information at different levels of abstraction and granularity, including specific facts, meta-data, and units of behavior [4].

In the context of MODM, interoperation among components is possible at many different levels of abstraction and granularity, ranging from factual information units (data objects) to meta-data (conceptual schema) to behavior. For this paper, we limit our investigation to the sharing of type objects, which we term *type-level* sharing. The sharing of individual instances (*instance-level* sharing) and the sharing of behavior or functions (*function-level* sharing) are examined in [4].

Figure 3: Partial conceptual schemas of two macromolecular databases

As shown in Figure 2, a special component termed the *sharing advisor* manages knowledge about existing (type) objects that components export; this knowledge resides in the *semantic dictionary*. The sharing advisor provides four "intelligent" services to the components of the federation: *Registration*, *Discovery*, *Semantic Heterogeneity Resolution*, and *Unification*. These services implement the discovery of (type) objects, and the folding of those objects into the environment of a local component. It is important to note that as it is nearly impossible to automate these processes, we provide substantial functionality and utilize guided user input as necessary.

4.1 Registration

Registration allows a (new) component to inform the sharing advisor about any information it is willing to share with other components in the federation. In a sense, it establishes an initial sharing context within the federation by logically connecting the exported information to the semantic dictionary via the sharing advisor. Incremental registration allows a component to augment its export schema with new information.

Figure 3 shows two partial conceptual schemas of component (scientist) A and component (scientist) B, respectively. Let us assume that component A has already registered the type objects **Protein Instances** and **Amino Acid Sequences** with the sharing advisor. This is reflected in the semantic dictionary shown in Figure 4a. When component B registers type object **Protein Structures**, the sharing advisor can use its existing knowledge (in this case the information obtained from component A) to determine if **Protein Structures** has any similarities with previously registered types, i.e., **Protein Instances** and **Amino Acid Sequences**. User interaction may be necessary to instruct the shar-

Figure 4: Evolution of a concept hierarchy in the semantic dictionary

ing advisor in case the information in the semantic dictionary is insufficient to detect (dis)similarities among type objects automatically. The newly acquired knowledge and the newly registered information are stored in the semantic dictionary for future consultation; see, for example, contents of the semantic dictionary in Figure 4b after the registration of the type object **Protein Structures**. In a sense, the semantic dictionary represents a dynamic federated knowledge base about sharable information in the federation.

In the remainder of this section, we illustrate how the sharing advisor can effectively utilize the semantic dictionary as a means of organizing various registered type objects in order to accommodate information sharing. We also introduce a set of sharing heuristics that guide the sharing advisor through registration.

4.1.1 The Semantic Dictionary

In the semantic dictionary, types determined to be similar by the sharing advisor are classified into a collection called a *concept*, within which subcollections called *subconcepts* can be further identified. This generates a *concept hierarchy*. Naturally, the relationships expressed in this concept hierarchy can only be approximations of the true, real-world relationships that exist between the exported types of different components. Additional mechanisms are needed to establish more exact relationships (see Section 4.3).

Figure 4 shows two snapshots of the concept hierarchy in the semantic dictionary taken at different times during the life-time of our example federation. Figure 4a indicates the concept hierarchy after types **Protein Instances** and **Amino Acid Sequences** of component A have been registered. Figure 4b shows the corresponding hierarchy after component B has registered the type **Protein Structures**. The hierarchy in Figure 4b

also indicates that **Protein Instances** and **Protein Structures** are representing similar information, since they belong to the same classification, called **Protein Information**[2]. In addition, the types **Protein Instances** and **Protein Structures** have properties that distinguishes one type from the other. Hence, they are also created as subconcepts of **Protein Information** in order to express their dissimilarities. By contrast, **Amino Acid Sequences** is similar to neither **Protein Instances** nor **Protein Structures**, and thus appears as a separate concept in the hierarchy. Similarity and dissimilarity of this kind is detected by the sharing advisor based upon sharing heuristics (described below), with user input as required.

The advantage of organizing type objects in a concept hierarchy is that a hill climbing technique can be used to place newly registered type objects. For example, consider a type object being registered which is determined to be dissimilar to members of concept **Protein Information** in Figure 4b. In this case, no further comparisons with members of subconcepts of **Protein Information** are necessary.

Generally, a type object represents a specific view of a corresponding real world concept, and is tailored to the focus and interest of the database component; therefore, the set of properties associated with the type object can be viewed as a subset of those associated with the real world concept. In Figure 3, type **Protein Instances** of component A, and **Protein Structures** of component B indicate two different views on the real world concept *protein*. In order to properly merge various views on a similar concept, the semantic dictionary is established in a bottom up fashion with the set of properties belonging to a concept at a particular level represented as the union of the properties of all its subconcepts. This is illustrated by concept **Protein Information** in Figure 4b. Using a concept hierarchy to organize exported types will incrementally establish a federated view of all the export schemas in the federation.

4.1.2 The Sharing Heuristics

The sharing heuristics employed draw upon the incremental clustering paradigm of machine learning, as described in [7]. The idea behind these heuristics is to assess the extent of the distinguishing capability of a property with respect to a concept; this allows the sharing advisor to determine if the meaning of a type object being registered can be determined based upon its properties, or whether further assistance from users is necessary. The distinguishing capability of a property with respect to a concept is based upon the *inter-concept dissimilarity* between concepts and the *intra-concept similarity* within a concept. As an example, consider property **code** of concept **Protein Information** in Figure 4b. This property has a high inter-concept dissimilarity, as no other concept at the same scope/level possesses such a property. On the other hand, **code** also has a high intra-concept similarity with respect to **Protein Information**, since this property is associated with all concept members of **Protein Information**, for example, **Protein Instances**. However, when looking at the subconcepts of **Protein Information**, property **code** has

[2]The name of this concept is automatically determined by a simple algorithm, which may be overridden by the user.

a low inter-concept dissimilarity with respect to each subconcept of **Protein Information**, since this property is possessed by all concepts within this same scope of a common superconcept, **Protein Information**. Note that a property does not necessarily possess the same extent of distinguishing capability across different levels.

Inter-concept dissimilarity and intra-concept similarity values are estimated via statistical analysis based on previously registered type objects. A statistical heuristic-based approach offers a degree of error resilience, allowing the accuracy of the distinguishing capabilities to be gradually improved over a period of time. A limitation of this approach is its potential oscillating nature during the early stages of a federation; however, this limitation becomes less significant as the number of components and type objects increases, damping the oscillation to a steady state.

4.2 Discovery

The purpose of our discovery mechanism is to identify appropriate information relevant to the request of a component initiating a sharing procedure. Although we have proposed a methodology to detect "similarities" among different types, it is not adequate in a federated environment where the goal is to integrate specific nonlocal information into the environment of a local component. This is because even though a remote object is similar to a particular local object, it might not be relevant within the intended context of a local component. Therefore, it is imperative that user characteristics of the local components be taken into consideration when locating relevant information. For this purpose, we have categorized three basic kinds of discovery requests, which when combined, allow a component to discover a wide variety of nonlocal information:

- Discovery Request Type 1—Similar Concepts.
  In this kind of discovery request, a component user is interested in locating type objects in remote components that are conceptually similar/related to a particular type object in the local component. All type objects belonging to the portion of the concept hierarchy in which the local type object resides (in the semantic dictionary) are appropriate to this request. For example, in the concept hierarchy of Figure 3b, **Protein Structures** of component B is a proper candidate to the request by component A for related information on **Protein Instances**.

- Discovery Request Type 2—Complimentary Information.
  In this kind of discovery request, a component user is interested in discovering additional information about a local type object. This may occur when component A is interested in additional information on **Protein Instances**, for example. All type members with different sets of properties belonging to the same portion of the hierarchy in which the local type object resides are proper candidates for this request. For example, **Protein Structures** of component B would also satisfy a request for additional information of **Protein Instances** issued by component A.

- Discovery Request Type 3—Overlapping Information.
  This kind of discovery request arises when a component is interested in locating non-local type objects that overlap in their information content with a component's local type. For example, component A would like to display all "protein"-like information using its own three dimensional viewing program which works on members of type **Protein Instances**. All types with similar properties as **Protein Instances** that belong to the sub-hierarchy rooted at **Protein Instances** are proper candidates for this request. According to the concept hierarchy of Figure 4b, there is no candidate type object that would satisfy such a request at this time.

4.3 Semantic Heterogeneity Resolution

After identifying relevant nonlocal information objects, a component may wish to fold them into its local information framework. However, the problem is to determine how this information can be unified with its own local data due to semantic discrepancies that may exist between related concepts in different components. Such semantic heterogeneity is a natural consequence of the independent creation and evolution of autonomous databases which are tailored to the requirements of the applications they serve [1, 13]. The purpose of our approach to semantic heterogeneity resolution is to resolve these discrepancies between the relevant nonlocal type object(s) and the local meta-data context (conceptual schema). This is a prelude to unifying the remote information into the local context. Our mechanism for resolving semantic heterogeneity employs a *(local) lexicon* for each component, which specifies its perspective on the precise relationship between its local types and a global set of commonly understood concepts. Specifically, knowledge is represented as a static collection of facts of the simple form:

<term> relationship descriptor <term>

A term on the left hand side of a relationship descriptor represents a local type object (i.e., the unknown), which is described by the term on the right side. We have an initial set of descriptors, which is extensible; we also anticipate collections of descriptors that are tailored to given application domains. The following is a list of basic conceptual relationship descriptors initially supported in Remote-Exchange:

R-Descriptors	Meaning
IDENTICAL	Two types are the same
EQUAL	Two types are equivalent
COMPATIBLE	Two types are transformable
KINDOF	Specialization of a type
ASSOC	Positive association between two types
COLLECTIONOF	Collection of related types
INSTANCEOF	Instance of a type
COMMON	Common characteristic of a collection
FEATURE	Descriptive feature of a type
HAS	Property belonging to all instances of a type

For example, we may have the following in the local lexicon of component A:

Protein Instances	KINDOF	**Protein Structures**	A protein instance is a specialization of protein structure.
"	COLLECTIONOF	**Proteins**	It is a collection of all proteins.
"	FEATURE	**Authors**	One of its characteristics is the existence of an author who discovered it.

The terms that are used to describe unknown concepts are taken from a dynamic list of concepts drawn from the semantic dictionary, characterizing the commonalities in a federation. Since interoperability only makes sense among concepts that model similar or related information, it is reasonable to expect a common understanding of a minimal set of concepts taken from the application domain. The semantic dictionary contains partial knowledge about all the terms in the local lexica in the federation, suggesting possible relationships between different terms. Utilizing this knowledge, each local lexicon describes the precise meaning of all type objects that the component exports. In our example (see Figure 4b), the set of commonly understood concepts at a particular moment could be:

$$\mathcal{C} = \{\textit{Amino Acid Sequences}, \textit{Authors}, \textit{Proteins}, \textit{Protein Information}, \textit{Protein Instances}, \textit{Protein Structures}\}$$

The essential idea of using a local lexicon is to represent the semantics of the shared terms in a more expressive and complete manner than in the conceptual schema. The additional semantic information is important for the following reason: as noted earlier, the results of the discovery process are not always conclusive enough to determine the exact meaning of a term, or more precisely, how two similar terms in different components are related. Similarly, it is not possible to "derive" the meaning/usage of terms by looking at their structural representation in the conceptual schema of a local component. In order to determine the relationships between objects in a federation, we realize that not

one single method but a combination of several different but complimentary approaches (i.e., Registration, Discovery, Semantic Heterogeneity Resolution) taken together is highly promising. For example, there is an important connection between local lexica and semantic dictionary. Local lexica contain only semantic information and no knowledge about any relationships among its objects; information that is necessary to solve the semantic heterogeneity problem. This kind of information is provided by the semantic dictionary (through registration) which contains partial knowledge about the relationships among all the terms in the local lexica in the federation. Note of course that the lexica and semantic dictionary are both dynamic, i.e., they grow (and shrink) as the amount of shared data in the federation increases (decreases).

The basic problem addressed by the semantic heterogeneity resolution mechanism may now be expressed, without loss of generality, as: given two objects, a local and a foreign one, return the relationship that exists between the two. Specifically, our strategy is based on structural knowledge (conceptual schema information) and the (known) relationships that exist between common, global concepts in set $\mathcal{C}$ and the two objects in questions (local lexicon, semantic dictionary). One characteristic of our approach is that the majority of user input occurs before the resolution step is performed (i.e., when selecting the set $\mathcal{C}$ and creating the local lexicon) rather than during.

4.4 Unification

At this point, the nonlocal object(s) can be unified with the corresponding local object(s). In some cases, the local meta-data framework (conceptual schema) must be restructured to achieve a result that is (1) complete, (2) minimal, and (3) understandable. Complete since the new, integrated schema must contain all concepts that were present before the unification process took place. Minimal since concepts should only be represented once. And understandable since the integrated schema should be easy to understand for the end user.

Upon importing the (meta-)data, structural conflicts with existing types in the component's local type hierarchy may arise. In [9], we enumerated in detail various conflicting possibilities that can arise while importing nonlocal meta-data into a local schema. Here, we illustrate our mechanism with the simple example of importing type **Protein Structures** of component B into A's schema. The following two possibilities exist:

- **Protein Structures** is (semantically) equivalent to **Protein Instances** in A's schema.
  In this case, we make **Protein Structures** a subtype of **Protein Instances**. Properties that previously belonged to **Protein Instances** remain there. Properties that belong to **Protein Structures** but not to **Protein Instances** are added to the new subtype. In those cases where **Protein Instances** has additional properties that do not exist for **Protein Structures** special null values must be assigned to all its instances (see Figure 5a). Note that "subsetting" is considered to be the basis for accommodating multiple user perspectives on comparable types used by most

Figure 5: A's schema after resolution and unification of "protein"-like information.

methodologies [1]. The case in which **Protein Structures** is identical to **Protein Instances** is a special case and only requires the importation of the type instances in which A is interested.

- **Protein Structures** is related to **Protein Instances** in A's schema.
  In this situation, a new supertype called **Proteins** is created that contains only the properties common to both types **Protein Instances** and **Protein Structures**. The properties in which **Protein Instances** and **Protein Structures** differ are associated with two subtypes which inherit the properties from their common supertype (see Figure 5b). Together, the new supertype and its two subtypes contain the same information as **Protein Instances** and **Protein Structures** in separate components before the unification [3].

Note that whether **Protein Structures** is related to **Protein Instances** or is equivalent to **Protein Instances** depends upon the perspective of the importing component, as specified in the local lexicon.

5 Conclusions and Directions

We have introduced a framework and mechanism for identifying and integrating type objects from diverse information sources. The mechanism is built on an object-based model containing features commonly found in most existing object-based database systems, and hence can be implemented in these systems with little or no modification to existing DBMS

software. We have also demonstrated the feasibility of our mechanism in a macromolecular environment, representing an area of tremendous and growing interest and importance [8, 20].

An important characteristic of our mechanism stems from the fact that our approach separates the discovery issue from the integration issue; prior work largely mixes the two, with the discovery issue implicitly a part of integration. We observed that different (albeit related) kinds of knowledge are needed in the discovery and integration processes; by identifying the knowledge needed in the different processes, a component is more likely to be able to access the most appropriate nonlocal information in the most natural way.

As noted, the ability of the sharing advisor to determine the similarity among type objects is critical to the success of our mechanism. As such, we are in the process of experimenting with the sharing advisor by measuring its behavior on diverse sets of test data. The measurements are based upon the following two metrics: (1) *completeness*, which measures if all the similar objects are being identified by the sharing advisor, and (2) *precision*, which measures if all the identified objects are similar.

We have focused here on the functionality, rather than performance efficiency. We are currently investigating the impact of "*lazy evaluation*" paradigm on reducing the amount of overhead involved in our system. Briefly, the "lazy evaluation" paradigm shifts the burden on determining the similarity among type objects from the registration process to the discovery process. In this way, the similarity among type objects is not determined unless the type object is actually used by a component.

Acknowledgements

The authors would like to acknowledge the very useful insights of the researchers involved in Remote-Exchange, including K.J. Byeon and Jonghyun Kahng. Special thanks to Kathryn Stuart who helped us understand the molecular biology that is used in our example scenario.

References

[1] C. Batini, M. Lenzerini, and S. Navathe. A Comparative Analysis of Methodologies of Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[2] S. Ceri and G. Pelagatti. *Distributed Databases: Principles and Systems.* McGraw Hill, 1984.

[3] U. Dayal and H. Hwang. View Definition and Generalization for Database Integration in Multibase: A System for Heterogeneous Distributed Databases. *IEEE Transactions on Software Engineering*, 10(6):628–644, 1984.

[4] D. Fang, J. Hammer, and D. McLeod. An Approach to Behavior Sharing in Federated Database Systems. In M.T. Özsu, U. Dayal, and P. Valduriez, editors, *Distributed Object Management*, pages 334–346. Morgan Kaufman, 1993.

[5] D. Fang, J. Hammer, D. McLeod, and A. Si. Remote-Exchange: An Approach to Controlled Sharing among Autonomous, Heterogenous Database Systems. In *Proceedings of the IEEE Spring Compcon*. IEEE, San Francisco, February 1991.

[6] P. Fankhauser and E. Neuhold. Knowledge Based Integration of Heterogeneous Databases. Technical report, Technische Hochschule Darmstadt, 1992.

[7] D. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, pages 139–172, 1987.

[8] K. Frenkel. The Human Genome Project and Informatics. *Communications of the ACM*, 34(11):41–51, 1991.

[9] J. Hammer and D. McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Intelligent & Cooperative Information Systems*, 2(1):51–83, March 1993.

[10] S. Hayne and S. Ram. Multi-User View Integration System (MUVIS): An Expert System for View Integration. In *Proceedings of the 6th International Conference on Data Engineering*. IEEE, February 1990.

[11] D. Heimbigner and D. McLeod. A Federated Architecture for Information Systems. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[12] M. Huhns, N. Jacobs, T. Ksiezyk, W. Shen, M. Singh, and P. Cannata. Enterprise Information Modeling and Model Integration in Carnot. Technical Report Carnot-128-92, MCC, 1992.

[13] W. Kent. The Many Forms of a Single Fact. In *Proceedings of the IEEE Spring Compcon*. IEEE, February 1989.

[14] W. Kent. Solving Domain Mismatch Problems with an Object-Oriented Database Programming Language. In *Proceedings of the International Conference on Very Large Databases*, pages 147–160. IEEE, September 1991.

[15] J. Larson, S.B. Navathe, and R. Elmasri. A Theory of Attribute Equivalence and its Applications to Schema Integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.

[16] W. Litwin and A. Abdellatif. Multidatabase Interoperability. *IEEE Computer*, 19(12):10–18, December 1986.

[17] F. Manola, S. Heiler, D. Georgakopoulos, M. Hornick, and M. Brodie. Distributed Object Management. *International Journal of Intelligent & Cooperative Information Systems*, 1(1):5–42, 1992.

[18] A. Mehta, J. Geller, Y. Perl, and P. Fankhauser. Computing Access Relevance to Support Path-Method Generation in Interoperable Multi-OODB. In *Proceedings of the International Conference on Very Large Databases*, pages 119–139. IEEE, August 1992.

[19] M. Papazoglou, S. Laufmann, and T. Sellis. An Organizational Framework for Cooperating Intelligent Information Systems. *International Journal of Intelligent & Cooperative Information Systems*, 1(1):169–202, 1992.

[20] J. Saldanha and J. Eccles. The Application of SSADM to Modelling the Logical Structure of Proteins. *CABIOS*, pages 515–524, 1991.

[21] A. Savasere, A. Sheth, S. Gala, S. Navathe, and H. Marcus. On Applying Classification to Schema Integration. In *Proceedings of IEEE 1st International Workshop on Interoperability in Multidatabase Systems*, pages 258–261. Kyoto, Japan, April 1991.

[22] A. Sheth, J. Larson, A. Cornelio, and S. B. Navathe. A Tool for Integrating Conceptual Schemata and User Views. In *Proceedings of the 4th International Conference on Data Engineering*, pages 176–183. IEEE, February 1988.

[23] T. Templeton, et al. Mermaid: A Front–End to Distributed Heterogenous Databases. In *Proceedings Intl' Conf. on Data Engineering*, pages 695–708. IEEE, 1987.

[24] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, March 1992.