Data Integration in the Large: The Challenge of Reuse

Arnon Rosenthal
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730 USA
arnie@mitre.org

Leonard J. Seligman
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102 USA
seligman@mitre.org

Abstract

Much published research on data integration considers only a "one shot" effort to produce an integrated schema or a multidatabase query. This paper examines a more complex environment. Our clients carry out multiple integration efforts, producing multiple kinds of integrated systems that involve overlapping subsets of their component databases. Metadata is costly to collect and maintain, so one wishes to reuse it wherever possible. We thus must devise ways to reuse integration metadata across integration efforts, though the efforts may have different goals and may concern overlapping subsets of the components. This paper identifies and examines issues of maximizing information and code reuse by organizations facing data integration in the large.

1 Introduction

Organizations increasingly need to build multidatabase systems that combine, exchange, or otherwise integrate information among independently developed databases and applications.[1] Our large customers realize that data integration will be costly, and require strategies for establishing a wide variety of integration arrangements, including integrated access, migration to standard interfaces, and intersystem dataflows. Communication is currently achieved either by expensive *ad hoc* code or by manual intervention at runtime. Current Computer Aided Software Engineering (CASE) tools are inadequate, because they focus on building and maintaining individual systems rather than multidatabase systems.

Much published research in this area concerns support for one-shot schema integration or integration via multidatabase query. Many methodologies ignore the other forms of integration. This paper examines a more complex environment in which organizations carry out multiple integration efforts, producing different kinds of integrated systems that involve overlapping subsets of its component databases. A major goal of this paper is to sensitize researchers to consider these typical complexities of customer environments.

Example: A customer may wish to establish a data flow that extracts recent intelligence data on a geographic region and ships it to a planning database of target information, at a different location. (Efficiency, security, and availability all argue for shipping the data to the downstream database, rather than providing integrated access to both systems.) Next, a display system desires integrated access to both the planner's target information and plans, plus a weather database and three-dimensional map data. Finally, the information may need to be made available to other systems, including those of allies, in a form that (as much as feasible) complies with standard or commonly used Air Force definitions.

We consider *reuse* – of metadata and of code – to be a major requirement for CASE tools for multidatabases. Tool vendors, when addressing multiple forms of integration, want to reuse code. User organizations want their metadata available in ways that maximizes reuse across multiple integration efforts. They also like the consistency that stems from code reuse.

Reuse should be considered even when designing a technique for an individual scenario, such as schema integration. Architectural overviews have often recognized that the same metadata may help in supporting multiple external views [Sheth90], but published

[1] The terms "integrate" and "multidatabase" are used broadly, to cover any way of establishing a system that handles data from multiple components, including loose coupling without a global schema. To limit the scope, we do not discuss transaction management or data model translations.

Proceedings of the 20th VLDB Conference
Santiago, Chile, 1994

methodologies are uneven in the reusability of their metadata products and their code modules. This unevenness may be unnecessary – major improvements can often be achieved by small changes to details.

Example: A query that derives a view spanning two components is difficult to reuse. First, it is large and monolithic, covering perhaps dozens of attributes. As a result, a simple change (e.g., to the derivation of one of the view's attributes) requires that a large, complex query be modified, instead of just a small modular chunk of knowledge. Second, information is tied to particular pairs of systems; assertions that related a component to a reference definition would be easier to reuse. Finally, it is difficult to combine partial information from multiple view definitions (e.g., to combine {views relating DB1 and DB2} with {views relating DB1 and DB3} to infer {views relating DB2 and DB3}).

Paper Roadmap: The contribution of the paper lies in establishing connections between the wider requirements and existing research, rather than in specific algorithms or new methodologies. We also wish to emphasize the importance of standardizing multisystem metadata to enable multidatabase CASE tools to interoperate.

A wide variety of data integration problems are described in Section 2. We briefly discuss the metadata needed for each plateau of functionality. Generally, analysts working on a particular form of integration would wish to be prompted for exactly the metadata needed for the chosen form. The discussion can thus guide builders of methodologies and tools in their modularization.

Section 3 provides some tactics that methodology-designers can use to select metadata that will be reusable. We identify tactics by which schema integration methodologies can produce metadata that is easy to reuse. Some of the listed techniques are described in the cited literature while others are our abstraction of implementation-level details of familiar methodologies, or our own adaptation. While few are novel, it seems important to consider them from the perspective of reuse.

Section 4 describes the central role of *repositories* in supporting information reuse. It argues that repositories could enable an industry of CASE tools for multidatabase systems. Standardized repositories would be particularly valuable since they would enable interoperation of separately-developed tools. This section also describes our efforts to develop a prototype repository designed to support data integration in the large.

2 Varieties of Data Integration Problems

2.1. Describing the Varieties of Integration

Much of the literature on multidatabase systems has emphasized one-shot schema integration and multidatabase queries. While these are important problems, several additional kinds of integration efforts are common in large organizations.

Recently, we analyzed the metadata requirements of data integration scenarios from MITRE's clients. We found a wide variety of scenarios. In addition, we found that it is common for a given component system to participate in multiple integration efforts. For example, a component system could be part of several federated views, could exchange data with other systems in a loosely-coupled interoperable multidatabase, and could be replaced by a new system that consolidates several existing systems.

We found that the following were typical goals of integration efforts in our clients' organizations:

- A new query interface to an existing system. In some cases the requirement may be to map the component to a new enterprise schema. In others, the changes may affect attributes but not class hierarchies. For example, one may need to provide a revised interface that attempts to use new preferred naming conventions, datatypes, and units for each attribute. My_Alt (in feet, as a string) might become Current_Altitude (in hundreds of feet, as an integer).
- A new interface (as above), but this time with support for update. DBMSs typically support some update operations against views; in the future they may also allow other operations (e.g., error messages in terms of view objects, administration of the underlying data). Metadata beyond that of a query interface is often required (e.g., to disambiguate updates).
- A query-only federated view of multiple existing systems. This is the most familiar product of schema integration.
- A more robust federated view, including support for updates, error processing, etc.
- Dataflows among systems, with facilities only for documenting the data interrelationships. In many current systems, these dataflows are implemented by manually-written extract and load programs in SQL and a 3GL. We are currently unscrambling a 6000 line program that extracts data from one Oracle database, converts it, and loads into another. The code combines data routing, attribute and class transformations, and constraint checking. Knowledge of intercomponent relationships is buried in these programs, and is not easily reused (e.g., to create a federated view).
- Dataflows among systems, with facilities for specifying and managing interdatabase consistency and other constraints [Rusinkiewicz91].
- Consolidation of multiple legacy systems into a new, more homogeneous system. In very large or newly-merged organizations, there are often multiple systems with roughly equivalent functionality but heterogeneous designs. The goal here is to understand the functionalities (via reverse engineering), to produce a new schema that supports the union of all important requirements, and then to implement a new system. Each legacy system will be replaced by a

copy of the new system; further integration efforts may proceed from this more homogeneous base.

- Evolution of a legacy multidatabase system . Several challenging scenarios involving migration of multiple databases and applications are described in [Brodie93]. A key requirement was that change would be introduced over a period of years, but service would continue throughout.

2.2 Overlap and Reuse of Required Metadata

The scenarios represent fairly natural plateaus of functionality. We examined the metadata that is needed to support each of them. The information might be of interest to tool builders, since it suggests natural modules of metadata to be captured. It also indicates the potential for very extensive reuse.

Figure 1 attempts to summarize the situation. Interestingly, the metadata that are useful for the most basic scenario (i.e., providing a new query interface to an existing system) are also needed for the other scenarios. For example, to support the development of a query-only federated view of existing systems, one could reuse virtually all of the information collected while providing a new, standards-compliant query interface. If the view is generated automatically from intercomponent assertions, the integrator needs to collect only the following additional information: semantic correspondences across the component systems and rules for identifying and handling multiple (possibly inconsistent) references to the same real-world instances. (If the default generated view is not the desired one, the deviation must also be documented.) Another example of metadata reuse among scenarios is provided by the Consolidation-via-Migration scenario, which requires basically the same information as the combination of the Migration and the Federated View scenarios.

The arrow from "Updatable Views" to "Data Flow among Loosely Coupled Interoperable Systems" indicates an interesting opportunity for reuse. In a typical system today, data flows into the receiving system via a manually-produced program that embeds database update requests in 3GL code. If instead, one defined a retrieval view of the source that matched an updatable view of the receiver, the transformation information would be much more declarative, and easier to reuse.

The overlap shown in Figure 1 is especially significant for large, complex organizations, in which many individual systems participate in multiple data integration efforts during their life cycle. If the metadata collected for one data integration effort could be stored in a repository and reused in subsequent efforts, that would result in substantial cost savings.

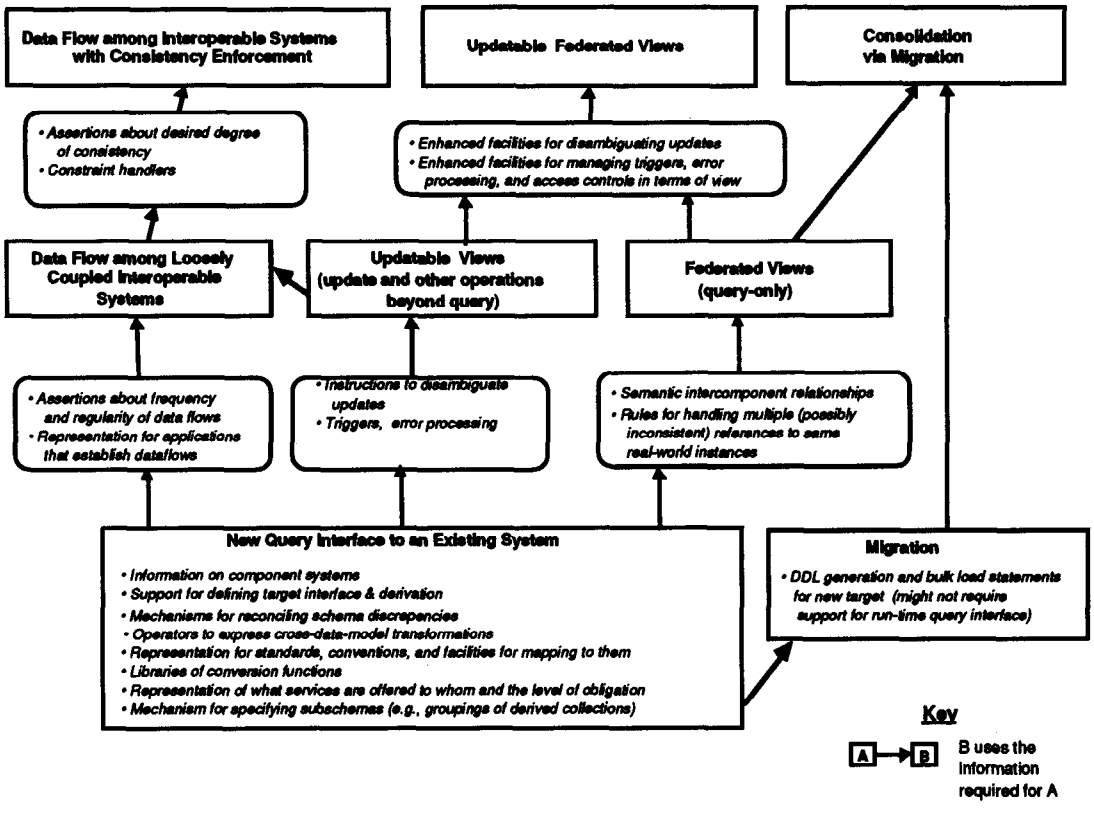

Figure 1. Overlapping Information Requirements Among Several Scenarios

3 Some Guidelines for Promoting Reusability

This section proposes some guidelines for methodology and tool developers to maximize the reusability of collected metadata. While few of these techniques are novel, it seems useful to gather and review them from the perspective of improving reuse of metadata.

3.1 Capture Small Modules of Information

View definitions are difficult to reuse, but the definitions can often be automatically generated from smaller modules of metadata. We give two examples.

A view that defines a new interface may involve many attributes, and the query that derives it may be a large expression in a textual query language. These are awkward units for reuse. For example, the next integration effort may require the same transformation on some of the attributes, and may require attributes not mentioned in the view.

Reuse will be easier if the information in the view definition is modularized. Frequently, the derivations of attributes in a view's target list are independent. The fact that Name is truncated to 20 characters is independent of the fact that Aircraft_Range is converted from miles to kilometers. Each of these can be stored in the repository as a single assertion, reusable as needed. An analyst who wishes to reuse or modify the conversion on Aircraft_Range can now avoid poring through a view definition that derives thirty other attributes. Security is also improved because one can release definitions selectively.

A similar tactic can be used with views that span components. As in [Sheth93], one can capture a collection of intercomponent assertions, and use them to generate the view. For example, suppose in separate integration efforts, a view that spans components 1 and 2 combines EMPLOYEE and WORKER, while a view that spans components 2 and 3 combines WORKER and PERSON. If a later integration effort on components 1 and 3 needs to combine information about EMPLOYEE and PERSON, it will be difficult to do so.

If one has intercomponent assertions instead of just views, the situation is easier. For example, one might have assertions on the actual populations, that every instance of EMPLOYEE in component 1 appears as an instance of WORKER in component 2, and that a similar inclusion holds between WORKER in component 2 and PERSON in component 3. Now one can infer an inclusion between EMPLOYEE and WORKER.

The example suggests an interesting open problem. Suppose one intends to integrate components A and B to form [AB], and then integrate [AB] with C. Assertions about [AB] are awkward to obtain (there are no experts in database [AB]) and to reuse. Can one instead express sufficient pairwise assertions among A, B, and C to enable the integration of [AB] with C? How many such assertions are needed?

3.2 Use Existing Reference Definitions: Enterprise Schemas and Standards

One powerful technique is to define an enterprise schema as a central point of reference. One then relates objects (classes, attributes, etc.) in each component schema to objects in this "standard" enterprise schema. For example, MCC's Carnot project [Collet91] used the Cyc knowledge base as a model for a large portion of "common sense reality." If objects in two component schemas are asserted to be "the same" as an object in the standard schema, then they are the same. More general intercomponent relationships can also be handled.

The reuse advantage is that to relate n systems one needs only n sets of intercomponent assertions, rather than $O(n^2)$. The information that one component system's class "Car" means the same as another component's "Voiture" is hard to reuse; the information that it matches the "standard" interpretation of Automobile is more helpful. Also, the enterprise schema may be well considered and documented, as it receives substantial attention. As a modest extension, one can provide multiple reference schemas instead of one. This enables integrators to refer to any "well known" object, even when there is no universally accepted conceptual schema.

A variant of reference schemas, widespread in U.S. Government standardization efforts, is the definition of generic attributes and value domains. For example, a database might include scores of attributes that denote a position (e.g., HomeBase, Target, Refueling). The definition of the notion of Position is captured once. Use of such generic definitions may deserve more emphasis in integration methodologies.

3.3 Create New Reference Definitions: Give Names to Common Properties

We cannot build a descriptive system that will describe (or even name) all shades of "meaning." However, whenever some property is given a name and a commonly understood meaning, there is substantial utility to placing this name and meaning as a reference definition, as in section 3.2. One can then assert the relationship between any schema object and the new reference definition. These definitions need not all be pushed into one schema, as that may require close cooperation.

For example, one might discover that attributes concerned with Speed have in their descriptions either a statement "absolute" or "relative." One then defines a new property name, say Frame_Of_Reference, with legal values "absolute" and "relative." Now when comparing the meanings of Speed attributes, one can easily check whether they have the same frame of reference. By capturing and naming such commonalities, we get the benefits of reference definitions. Units, Datatype, and Scale are other properties one might define.

Sets of reference definitions offer more flexibility but less uniformity than having an enterprise schema at the center. Avoidance of homonyms is critical: if two components say VolumeUnit="gallon", chaos will result if one uses British gallons and the other American. It is also desirable to recognize synonyms, as this allows more similarities to be discovered. [Sciore94] uses sets of (property name, property value) pairs to drive a mediator that determines whether an application can use a database's data. (If the data is not directly usable, the mediator attempts to find a conversion function that alters the offending property.)

Assertions with respect to reference definitions are convenient for reuse. An assertion that "Database1.Aircraft.Range.LengthUnit=miles" will be useful whether integrating with Database2 or Database3. In contrast, an assertion "Database1.Aircraft.Range and Database2.Aircraft.MaxDistance are in compatible units" will be of less use when confronting Database3.

3.4. Move Individual components Toward the Desired Standard

When one produces a schema to integrate two components, the metadata will be more reusable if one splits the integrating view definition into two stages. The first stage transforms the individual schemas, providing each with an interface that better conforms to outside practice and to each other. For example, one might transform attributes' units, scale, precision, and datatypes in one or both component schemas, to make them match. One can perform joins and other operations to bring relations into closer correspondence.[2] The views of the components now exhibit less diversity than the original components. This should simplify later integration efforts.

Even if the immediate goal is schema integration, there are other benefits to single-component views that better conform to an organization's standards. For example, due to differing requirements and decentralized development, the European, Pacific, and other regional databases do not have identical schemas. Suppose that after schema integration is completed, one wishes to port a planning or display application currently running against the Pacific database so that other regions will be able to run the application over their own data.. The port will be easier if the components' interfaces have been made less diverse.

[2] The alternative is to assert correspondences between dissimilar structures in the two components. This approach has the disadvantage of duplicating much of the transformational power of the query language in the intercomponent assertion language.

4 Toward Repositories for Data Integration in the Large

The previous section describes guidelines for maximizing metadata reuse across multiple data integration efforts. In order to achieve this reuse, however, organizations need a mechanism for managing huge amounts of metadata required for data integration in the large. This section argues that metadata repositories for multidatabase systems provide such a mechanism and that they are a critical enabler for an industry of CASE tools for multidatabases. In addition, we describe our efforts to prototype such a repository.

4.1 The Critical Role of Repositories

The cost of developing and evolving multidatabase systems could be reduced by extending software engineering repositories [Bernknopf90] to support such systems. The primary benefits of using such a metadatabase are the following:

- It provides a critical enabling technology which would permit CASE companies to attack data integration problems incrementally, via tools that each incorporate some useful expertise.
- It would permit substantial information reuse, by exploiting the commonalities in the metadata gathering requirements of the different data integration scenarios. This is especially important in large complex organizations, in which a given component system frequently participates in multiple data integration efforts.

4.1.1 Discussion

Commercial heterogeneous DBMS products (e.g., Uniface, Oracle version 7) can provide transparent access to multiple systems despite differences in platforms, operating systems, and DBMSs. But (except for DII's Interviso) they do little to identify correspondences and discrepancies between component systems, resolve the discrepancies between them, incrementally define a federated view (or a new target schema), create the mappings from the component systems to the federated view (or target schema), and specify and manage constraints across multiple systems. We need CASE tools that help build and manage a metadatabase about multidatabase systems.

The extensive literature on schema integration, federated databases, and multidatabase interoperability addresses these issues (e.g., [Batini86, Sheth90, Litwin90, Wiederhold92]). While researchers have investigated the data integration process and have made progress on supporting individual integration tasks, two things have made it difficult to transition their results. First, each individual effort has naturally addressed only a narrow part of the integration problem. Second, the tools that have emerged from these efforts have been closed, unable to interoperate to form a greater whole. There has been

insufficient attention to developing a framework that would connect a large number of niche tools.

Example: To illustrate the benefit of getting multiple tools to interoperate, consider the following plausible scenario. First, reverse engineering tools are used to help capture information about one or more component systems in a common (possibly semantic) data model. Next, another tool is used to hypothesize semantic relationships between pairs of attributes from multiple systems. One can then imagine the integrator turning to a variety of other tools to perform subsequent phases of the integration process. For example, a tool which uses attribute assertions to infer relationships among classes, such as that described in [Sheth93], might be useful here. Tools which analyze instance-level data in order to hypothesize relationships across systems, such as [Beck92] or [Li94], could also be employed. Tools which use declarative specifications about interdatabase consistency requirements to automatically generate consistency enforcement procedures, such as [Ceri93] and [Seligman93], provide further examples.

To date, work on repositories and Integrated CASE (I-CASE) [Chen92] has focused on the development of single systems. In I-CASE systems, multiple specialized tools interoperate by exchanging information through a common repository. While repository standards (e.g., IRDS, PCTE) have received very limited vendor support, cottage industries of specialized CASE tools are emerging, centered around the repositories of the major CASE vendors. However, the repository-centered I-CASE approach has yet to be applied to CASE for developing and evolving multidatabase systems.

We contend that the development of repositories and repository standards for multidatabases is a critical step toward enabling a new industry of data integration CASE tools. The existence of a common infrastructure would support interoperability of specialized integration tools. Individual tools could be added relatively easily, facilitating the transition of niche tools and techniques from the research community to government and industry. Users would benefit by not being confined to a single vendor. In addition, vendors of integrated systems could acquire tools from multiple sources.

While to date there has been limited practical experience with using and managing multidatabase repositories, there may be advantages to standardizing their schemas before there are many conflicting proprietary variants.

4.2 A Repository Prototype

We are currently working to define requirements for a repository for data integration in the large, to develop a schema for such a repository, and to refine that schema through experimentation with a prototype system. We have restricted our focus to those aspects of the repository that are unique to the development and evolution of multidatabase systems; we have not addressed issues that also exist for single-system repositories (e.g., versioning, control and presentation integration [Chen92]). In addition, we have focused on issues of data heterogeneity (i.e., representation and semantics) and not on infrastructure heterogeneity (i.e., heterogeneous networks, operating systems, data models, and DBMSs), all of which are receiving considerable attention in the commercial marketplace.

Some of the issues being addressed by our work are:

- What metadata representations are best suited to supporting reuse across diverse integration efforts? We seek to maximize reuse across the different integration scenarios described in Section 2, as well as across efforts employing different integration strategies (e.g., bottom-up, top-down, and hybrid approaches [Sheth90]).
- What are the core reusable modules which ought to be part of the repository infrastructure? What functionality belongs in tools, and what in the repository?
- What constructs are necessary to support the incremental specification of an integrated or interoperable system? This requires facilities for a tool or a human to make assertions about partial, uncertain, and negative information.
- What kinds of background knowledge are useful to multiple integration tools and how should it be represented? Examples include enterprise schemas, generic data elements, knowledge about naming standards and conventions, and both domain-specific and domain-independent knowledge bases.

We have developed an initial prototype of a multidatabase repository using the ITASCA object database; for prototyping, the convenience of developing over an OODBMS outweighed conformance to standards. We have used the repository to capture the metadata required to support transparent access to two autonomous law enforcement databases, and are refining the repository based on that experience.

In the coming months, we will be using the prototype to manage metadata for performing different kinds of data integration efforts using overlapping subsets of several component systems. In addition, another MITRE group, which is researching techniques for automatically generating code for intersystem dataflows from a declarative specification, is developing a relational adaptation of our schema. These efforts should provide valuable lessons in our efforts to provide better support for managing and reusing metadata for data integration in the large. In addition, demonstrations of the prototype may help convince procurement officials to require contractors to deliver integration metadata for any new or reengineered system.

5 Conclusions

We discussed how organizations perform multiple, overlapping integration projects, often involving

overlapping sets of data. We indicated that reusability of metadata is an important issue for user organizations, and reusability of tool code is desirable. For CASE tool vendors, reuse of code modules is important.

We identified numerous varieties of integration. While the individual problems are not new, researchers rarely refer to the whole collection. Modules of functionality were identified and placed in a hierarchy.

We then argued that reuse should be considered whenever a methodology for any kind of integration is developed. We found both good and bad practices in published algorithms. Some guidelines to make metadata and functionality more reusable were given. Once again, our contribution was to supply the necessary context and abstract the problem, rather than to provide new techniques.

Finally, we argued that a metadata repository for multidatabase systems is a critical enabler of information reuse and of an industry of multidatabase CASE tools. We briefly described a prototype that we are in the process of building.

Acknowledgements

This work has been supported by MITRE Sponsored Research . It has benefited greatly from interactions with the Distributed Object Management Integration System and the Data Interoperability Between C3I Systems projects, respectively sponsored by Rome Laboratory C3AV and by ESC/XR,ENS. The authors would also like to thank Chris Bosch for his helpful comments and his leadership of the prototyping efforts.

References

[Batini86] C. Batini, M. Lenzerini, and S. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM ComputingSurveys*, December 1986.

[Beck92] H. Beck, T. Anwar, and S. Navathe, "Classification Through Conceptual Clustering in Database Systems," *International Conference on Information and Knowledge Management (CIKM-92)*, Baltimore, MD, November 1992.

[Bernknopf90] J. Bernknopf, "Repository Race Getting Crowded," *Software Magazine*, July 1990.

[Brod93] M. Brodie, M. Stonebraker, "DARWIN: On the Incremental Migration of Legacy Information Systems," TR-0222-10-92-165, GTE Laboratories, Waltham, MA 02254.

[Ceri93] S. Ceri and J. Widom, "Managing Semantic Heterogeneity with Production Rules and Persistent Queues," *Proceedings of International Conference on Very Large Databases (VLDB)*, August 1993.

[Chen 92] M. Chen and R. Norman, "A Framework for Integrated CASE," *IEEE Software*, March 1992.

[Collet91] C. Collet, M Huhns, W Shen, "Resource Integration Using a Large Knowledge Base in Carnot," *IEEE Computer*, Vol. 24, No. 12, December 1991.

[Guha90] R. Guha and D. Lenat, "Cyc: A Midterm Report," *AI Magazine*, Vol. 11, No. 3, Fall 1990.

[Li94] W. Li and C. Clifton, "Semantic Integration in Heterogeneous Databases Using Neural Networks," *VLDB*, September 1994.

[Litwin90] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, Special Issue on Heterogeneous Databases, Vol. 22, No. 3, Sept. 1990.

[Rusinkiewicz94] M. Rusinkewicz, A. Sheth, G. Karabatis, "Specifying Interdatabase Dependencies in a Multidatabase Environment," *IEEE Computer*, Vol. 24, No. 12, December 1991.

[Sciore94] E. Sciore, M. Siegel, A. Rosenthal, "Using Semantic Values to Facilitate Interoperability among Heterogeneous Information Systems," to appear in *ACM Transactions on Database Systems*, 1994

[Seligman93] L. Seligman and L. Kerschberg, "Knowledge-base/Database Consistency in a Federated Multidatabase Environment," Research Issues in Data Engineering: Interoperability in Multidatabase Systems (RIDE-IMS '93), Vienna, Austria, April 1993.

[Sheth90] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, September 1990.

[Sheth92] A. Sheth and V. Kashyap, "So Far (Schematically) yet So Near (Semantically)," *Proceedings of IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems*, DS-5, Lorne, Victoria, Australia, November 1992.

[Sheth93] A. Sheth, S. Gala, and S. Navathe, "On Automatic Reasoning for Schema Integration," *International Journal on Intelligent and Cooperative Information Systems*, Vol. 2, No. 1, March 1993.

[Wiederhold92] G. Wiederhold, "The Roles of Artificial Intelligence in Information Systems," *Journal of Intelligent Information Systems*, August 1992.