The Semantic Integration of Information Models

Michael N. Huhns and **Munindar P. Singh**
Microelectronics and Computer Technology Corporation
Information Systems Division
3500 West Balcones Center Drive
Austin, TX 78759-6509
(512) 338-3651 or huhns@mcc.com

Abstract

This paper describes a method for integrating separately developed information models. The models may be the schemas of databases, frame systems of knowledge bases, or process models of business operations. The method achieves integration at the semantic level by using an existing global ontology to resolve inconsistencies. The integrated models provide a coherent picture of an enterprise and enable its resources to be accessed and modified coherently. The method achieves integration at the semantic level by using an existing global ontology to resolve inconsistencies. Some heuristics are presented that may be used along with limited input from humans to guide this process.

Dimensions of Semantic Integration

Today's corporate computing environments are heterogeneous, containing many independent information resources of different types, such as a database management system with its databases, an expert system with its knowledge base, an information repository, or an application program. Unfortunately, the resources are often mutually incompatible in syntax and semantics, not only due to the different types, but also due to mismatches in underlying hardware and operating systems, in data structures, and in corporate usage. Information resources attempt to model some portion of the real world, and necessarily introduce simplifications and inaccuracies in this attempt that result in semantic incompatibilities.

These semantic incompatibilities must be resolved [Sheth and Larson, 1990] for the following pressing reasons:

1. Applications that span several of the resources must operate correctly.
2. A coherent picture of the enterprise is often needed for decision making and efficient business operations.
3. Applications must interoperate across a global enterprise. This is especially important since strategic business applications that require intercorporate linkage, e.g., linking buyers with suppliers, or intracorporate integration, e.g., producing composite information from engineering and manufacturing views of a product, are becoming increasingly prevalent.

One resolution would be to construct a homogeneous environment, but this is impractical. Instead, we utilize enterprise modeling and model integration to yield the appearance and effect of homogeneity.

Enterprise modeling is a corporate activity that produces models of the information resources, information flows, and business operations that occur in an enterprise. In many cases, models of information resources are already available. For example, the information present in a database is modeled by the schema for the database. Similarly, the information present in an object-centered knowledge base is modeled by the ontology of the objects. The goal of our semantic integration research has been to develop a method for integrating such existing models that overcomes the above incompatibilities, yielding coherency and consistency.

The method provides logical connectivity among information resources via a semantic service layer that automates the maintenance of data integrity, and provides an approximation of global data integration across systems, thus enabling the resources to be used coherently. This layer is a fundamental part of the Carnot architecture [Cannata, 1991], which provides tools for interoperability across global enterprises.

Two approaches have been suggested in the literature for integrating heterogeneous databases [Buneman *et al.*, 1990]:

- The *composite approach* introduces a global schema to describe the information in the given databases. Users and applications are presented with the illusion of a single, centralized database. Explicit resolutions are specified in advance for any semantic conflicts among the databases. However, the centralized view may differ from the previous local views and existing applications may not execute correctly any more. Further, a new global schema must be constructed every time a local schema changes or is added.

- The *federated* [Heimbigner and D. McLeod, 1985] or the *multidatabase* [Litwin *et al.*, 1990] approach presents a user with a collection of local schemas, along with tools for information sharing. The user resolves conflicts in an application-specific manner, and integrates only the required portions of the databases. This approach yields easier maintenance, increased security, and the ability to deal with inconsistencies. However, a user must understand the contents of each database to know what to include in a query: there is no global schema to provide advice about semantics. Also, each database must maintain knowledge about the other databases with which it shares information, e.g., in the form of models of the other databases or partial global schemas [Ahlsen and P. Johannesson, 1991]. For n databases, as many as $n(n \Leftrightarrow 1)$ partial global schemas may be required, while n mappings would suffice to translate between the databases and a global schema.

We base our methodology on the composite approach, but make four changes that enable us to combine the advantages of both approaches while avoiding some of their shortcomings. First, we use an existing global schema—the Cyc knowledge base [Lenat and Guha, 1990]. The schemas of individual resources are compared and merged with Cyc but not with each other, making a global schema much easier to construct and maintain. Using Cyc is significant, because of

1. its size: it covers a large portion of the real world and the subject matter of most information resources,
2. its rich set of abstractions, which ease the process of representing predefined groupings of concepts,
3. its knowledge representation and inference mechanisms, which are needed to construct, represent, and maintain a global schema, and
4. its typing mechanism, which is used to integrate and check the consistency of query results.

Second, unlike most previous work on integration, we use not just a structural description of the local schemas, but all available knowledge, including

1. schema knowledge, i.e., the structure of the data, integrity constraints, and allowed operations;
2. resource knowledge, i.e., a description of supported services, such as the data model and languages, lexical definitions of object names, the data itself, comments from resource designers and integrators; and
3. organization knowledge, i.e., the corporate rules governing use of the resource.

Third, we capture the mapping between each individual resource and the global schema in a set of *articulation axioms*: statements of equivalence between components of two theories [Guha, 1990]. The axioms provide a means of translation that enables the maintenance of a global view of all information resources and, at the same time, a set of local views that correspond to each individual resource. An application can retain its current view, but use the information in other resources. Of course, any application can be modified to use the global view directly to access all available information.

Fourth, we consider knowledge-based systems (KBSs) as well as databases. It should be clarified that we are considering KBSs rather than mere knowledge bases. A knowledge base is just a database, perhaps with a rich frame hierarchy. However, a KBS is a system that properly contains a knowledge base and a means, typically consisting of rules and an inference engine, for operating on the knowledge. It is largely irrelevant to the discussion in this paper whether the rules themselves are first class objects. How such a KBS is integrated depends on which of the following three roles it plays in an information system and would play in the resulting integrated system.

- **The KBS as an Information Resource:**
  The KBS simply contains information and responds to queries about it. Some of this information may be explicit in its knowledge base but, unlike a database, there may also be some implicit information that is made explicit only on demand. This is the weakest case of KBS integration, since no assumptions are made about the applications of which the given KBS is a resource. Thus, in this case, the KBS is integrated in approximately the same way as a database. Even so, this case is interestingly different because the KBS is not simply an information store—it also contains rules. These rules can aid the process of integration.
- **The KBS as a Driver:**
  Here, the KBS is the main application, and integration of the other available resources is done relative to this application. In this case, the KBS maintains knowledge about the other resources and about how to access them. That is, the KBS has information about the schemas of the other resources and the actual predicates or relation names so that it can invoke them without further translation. Also, rules in the KBS should fire off of all appropriate databases directly. Thus, the global schema required depends on the KBS; our approach aims to capture much of this dependency.
- **The KBS as a Federating Mechanism:**
  In this case, the KBS serves semantically as an intermediary between applications and databases. The KBS receives queries and updates and converts them to appropriate queries and updates on the views it integrates. The KBS maintains representations of the models of the databases in its charge. It embodies the specifications of appropriate conflict resolution algorithms, and specifications of how to dis-

tribute subqueries among databases and merge the results. This case differs from that of the KBS as a driver, because there is no unique application here and none of the applications may be implemented in the KBS itself.

Semantic Transactions with Global and Local Views

As shown in Figure 1, a resource is integrated by specifying a syntax and a semantics translation between it and the global schema. The syntax translation provides a bidirectional translation between a local data manipulation language, DML$_i$, and a global context language, GCL, which is based on extended first-order logic. The semantics translation is a mapping between two expressions in GCL that have equivalent meanings. This is accomplished by a set of articulation axioms, having the form $ist(G\ \phi) \Leftrightarrow ist(S_i\ \psi)$ where ϕ and ψ are logical expressions and *ist* is a predicate that means "is true in the context." This axiom says that the meaning of ϕ in the global schema G is the same as that of ψ in the local schema S_i. At most n sets of axioms are needed for n resources.

After integration, one can access the resources through the global view, which gives the illusion of a single information resource, but requires that GCL be used. Queries and updates can also be issued against a local view. In that case, they are first translated into GCL and then into different DML$_i$ and distributed to appropriate information resources. Thus, applications need not be modified to access the extra information that becomes available.

To illustrate the idea, we describe how transactions are processed semantically through the global and local views of two integrated databases. The two databases have the same domain (hotels), but use different data models. The Fodor database, shown in Figure 2, uses an object-oriented data model. A hotel is represented as a class called `FodorInfo`. The features of hotel are represented as fields of the class or as other object classes pointed at by `FodorInfo`. The AAA database, shown in Figure 3, uses the relational model. A hotel is called `AAAInfo` and represented as a relation. The features of hotel, such as name and address, are represented as columns. Note that these schemas represent different perspectives and different information about hotels.

Some of the Cyc concepts used in integrating these databases are the collections `Lodging` and `Restaurant`, and the predicates `hasAmenities`, `phoneNumber`, and `instanceOf`. Articulation axioms that map between the two database schemas and the global schema include

ist(G instanceOf(?H Lodging))$\Leftrightarrow$
  ist(AAA instanceOf(?H AAAInfo))
ist(G phoneNumber(?H ?P))$\Leftrightarrow$*ist(AAA phone(?H ?P))*
ist(G hasAmenities(?H ?F))$\Leftrightarrow$*ist(AAA facility(?H ?F))*
ist(G instanceOf(?H Lodging))$\Leftrightarrow$
  ist(Fodor instanceOf(?H FodorInfo))
ist(G hasAmenities(?H ?F))$\Leftrightarrow$
  ist(Fodor facility(?H ?X)$\wedge$*facilityCode(?X ?F))*

Based on its local view of the AAA database, an application might issue the following query for the phone numbers of hotels that have a restaurant:

```
SELECT phone FROM AAAInfo
            WHERE facility = "Restaurant"
```

This local SQL query is first translated into GCL by the SQL-GCL syntax translator:

```
instanceOf(?L AAAInfo) ∧
instanceOf(?R Restaurant) ∧
facility(?L ?R) ∧
phone(?L ?P)
```

This expression is then mapped by articulation axioms into a new expression whose semantics is meaningful in the global schema:

```
instanceOf(?L Lodging) ∧
instanceOf(?R Restaurant) ∧
hasAmenities(?L ?R) ∧
phoneNumber(?L ?P)
```

This is then translated into different local queries using the appropriate articulation axioms in reverse. The translation for the Fodor local schema is

```
instanceOf(?L FodorInfo) ∧
facilities(?L ?F) ∧
facilityCode(?F ?R) ∧
instanceOf(?R Restaurant) ∧
phone(?L ?P) ∧
phoneNum(?P ?N)
```

These queries are then translated into appropriate DML$_i$ before being sent to the databases. For example, the query sent to the Fodor database is the following object-oriented expression (using ITASCA^{TM} syntax):

```
(SELECT (FodorInfo phones phoneNum)
        (= (path (some facilities)
                 (some facilityCode))
           "Restaurant"))
```

After the transactions are executed, the distributor assembles the results in the local view. In this example, the result is a column of phone numbers, because the local view of the AAA database is in the relational model. Note that these phone numbers come from two databases and the list may be much longer than that from the AAA database alone. However, users and applications need not be aware of the extra source of information.

The Development of Articulation Axioms

The articulation axioms for an information resource are developed in three phases that are detailed in sub-

Figure 1: Global and local views in semantic transaction processing

sequent subsections:

1. *schema representation*, in which a Cyc context containing a model for the resource is produced,
2. *concept matching*, in which concepts from the model are matched with concepts in Cyc's base context—the global schema, and
3. *axiom construction*, in which the matches are converted automatically into articulation axioms by instantiating templates for these axioms with terms from the matches.

The matching phase requires human interaction: frames may have to be created in the global schema and the model of the local schema may have to be augmented with additional properties (semantics) to ensure a successful match.

Schema Representation

In this phase, we represent the given schema as a set of frames and slots in a Cyc context created specially for it. These frames are instances of frames describing the data model of the schema, e.g., (for a relational schema) `Relation` and `DatabaseAttribute`.

We define three types of frames for representing schemas:

1. `DatabaseSchema` frames, describing the schemas for different data models,
2. `DatabaseComponent` frames, describing the major components of schemas, such as relations and entities, and
3. `DatabaseLink` frames, describing different kinds of links used to refine and relate the major components.

Every schema and every one of its components (re-

Figure 2: The object-oriented schema for the Fodor database

RELATIONS	COLUMNS
AAAInfo	name* address rateCode lodgingType phone facility
AAADirection	address* direction
AAACredit	name* creditCard*
AAARate	name* season* 1P 2P1B 2P2B XP fCode

Figure 3: A relational database schema for the AAA Tour Book database

lation, attribute, etc.) is an `instanceOf` these types and belongs to a context characterizing that schema. The slot `dBSchemaMt`, defined for `DatabaseSchema`, is used to express the relationship between an instance of a schema and its context. Information about the usage of a resource and the functionalities it provides are represented similarly, i.e., using frames such as `RelationalService`, `ERService`, `RelationalDDLType`, and `ERTransactionType`. Examples of the functionalities are Data Description Language (DDL), Data Manipulation Language (DML), and transactions.

Matching

How articulation axioms are generated and how different schemas are integrated depends crucially on the process of matching them. For resource integration, the problem of matching may be framed as follows: given a representation for a concept, find its corresponding concept in the global schema. As remarked above, in our work, both the representation for the given concept and the global schema are represented in Cyc. This makes the problem tractable; even so, the problem is difficult. There are several factors that affect this phase: there may be a mismatch between the local and global schemas in the depth of knowledge representing a concept, and there may be mismatches between the structures used to encode the knowledge. For example, a concept in Cyc can be represented as either a collection or an attribute [Lenat and Guha, 1990, pp. 339ff].

If the global schema's knowledge is more than or equivalent to that of the local schema's for some concept, then the interactive matching process described in this section will find the relevant portion of the global schema's knowledge. This knowledge will be in one of Cyc's two forms for concept representation. If the global schema has less knowledge than the local schema, then knowledge will be added to the global schema until its knowledge equals or exceeds that in the local schema; otherwise, the global schema would be unable to model the semantics of the resource. The added knowledge refines the global schema; as more and more schemas are integrated, the global schema grows. This constrains matches with schemas that are considered later: thus the proposed matches improve in quality.

Lexical Heuristics

Finding correspondences between concepts in the local and global schemas is a subgraph-matching problem. We base subgraph matching on a simple string matching between the names or synonyms of frames representing the database schema and the names or synonyms of frames in the global schema. Matching begins by finding associations between attribute/link definitions and existing slots in the global schema. After a few matches have been identified, either by exact string matches or by a user indicating the correct match out of a set of candidate matches, possible matches for the remaining schema concepts are greatly constrained. Conversely, after integrating an entity or object, possible matches for its attributes are greatly constrained.

Let a_{ij}, where $j = 1, 2, \ldots, n$, denote the attributes of concept E_i in a local schema. E_i is the domain of the attributes, i.e., the entity, relationship, relation, class, or object for which the a_{ij} are defined. Let s_j be the global schema slot that corresponds to, or matches, a_{ij}.

Observation 1 *The domain C_j of slot s_j is a generalization of the concept in the global schema that matches E_i.*

For example, the domain of each of the attributes, `facility` and `phone`, is the entity `AAAInfo`, whereas the domains of the corresponding Cyc slots, namely `hasAmenities` and `phoneNumber`, are the frames `HumanOccupiedStructure` and `Agent`, respectively. These are generalizations of Cyc's `Lodging`, which is the frame whose semantics most closely corresponds to `AAAInfo`.

As we match each of the attributes of E_i, we compute the common subdomain of the domains of their corresponding slots. The resulting common subdomains, although still generalizations of E_i, approximate it more and more closely.

Observation 2 *The "best" match for E_i is $\bigcap_{j=1}^{n} C_j$, the most general common subdomain (greatest lower bound in the generalization hierarchy) of the slot domains.*

In the above example, the most general common subdomain of `HumanOccupiedStructure` and `Agent` is `ServiceOrganization`, a generalization of `Lodging`. This would be suggested as the approximate match for `AAAInfo`. If no other attributes are matched, this would also be the *best* match that could be determined automatically for `AAAInfo`.

The greatest lower bound might not exist as a single frame in the global schema, however; it might be a set of frames. For example, the greatest lower bound in the above example would be the set {`HumanOccupiedStructure Agent`} if the frame `ServiceOrganization` did not exist. In such a case, a frame would be created in the Cyc knowledge base with the frames in the set computed above listed as its generalizations.

Heuristics Based on Rules

While databases are passive, KBSs are active. Intuitively, the rules of a KBS capture what we might call the "operational semantics" of the representations in the underlying knowledge base. In other words, it is the rules of the KBS that tie the entities represented in the knowledge base with each other and with meanings in the domain. The rules give a semantics to those representations by relating them to the real world, as it were. For example, an internal representation, `Broker`, can be taken to be a real broker because it participates in appropriate ways in certain financial transactions. Rules can thus be exploited in different ways in the matching phase.

Using heuristics based on rules for matching concepts helps limit the set of possible matches significantly and improves the chances of obtaining a match that preserves the important semantic properties of the application. It also reduces the amount of input required from human designers, thereby making it simpler for them to guide the process of integration. Thus the added complexity of dealing with a KBS instead of a database has immediate pay-backs in terms of the effort required for integration and the quality of the solution. In this abstract, we describe three of the simplest ways in which rules can be used for matching underlying concepts.

- **Syntactic type checking:**

  This is intuitively the most obvious of the heuristics. As an abstract example of its application, consider the rule

  ```
  If foo(?x ?y) and bar(?y ?z)
   then baz(?x ?y ?z)
  ```

  By inspection of the relations or predicates occurring in this rule and the variables that occur in different argument positions in it, we can determine constraints, such as the following, on potential mappings of the predicates `foo`, `bar`, and `baz`. At the very least, the type of the first argument of `foo` must be compatible with the type of the first argument of `baz`; this is because if the rule ever fires, `?x` must be bound to something and that must, therefore, be a legal argument to both `foo` and `baz`. Similarly, the types of the second argument of `foo` and of the first argument of `bar` must be compatible. This must be so, even if the actual instances of these relations are such that the rule would not fire on them.

  Thus the above rule necessarily precludes matches in which `foo` and `bar` are matched with entities in Cyc that are incompatible in the appropriate sense. In a way, this heuristic is quite weak: it yields only claims that are logically valid.

- **Constraints on domains and ranges:**

  While syntactic type checking is a useful heuristic, it is often too cautious. Fortunately, it can be easily improved in a large number of cases. The improvement is based on the intuition that rules are usually written for the entire concepts involved, not for some arbitrary subclasses of them. That is, the restrictions on the firing of a rule are explicit in its antecedent. This means that, in terms of the preceding example, the type of `foo` must not just be compatible with the type of the first argument of `baz`, it must be a subclass of the latter. It is easy to see that this is a major improvement in the case of concept hierarchies that allow multiple inheritance. Such hierarchies are, of course, quite the norm. This heuristic prevents lots of spurious concepts corresponding to the intersections of the given concepts from being created.

As a concrete example, consider a KBS containing a rule involving `Broker` and `Client`, where the rule refers to the relation of `isAdvisorOf` holding between an instance of `Broker` and an instance of `Client`. Thus, in the matching process, we consider the binary relation `isAdvisorOf`, and the concepts, `Broker` and `Client`. The class `Broker` must be matched to a subclass of the domain of `isAdvisorOf` and `Client` to a subclass of its range. This is a substantive claim—just asserting that `Broker` and `isAdvisorOf` are compatible would mean that their greatest lower bound was not identical to the intensionally empty concept, which is not much help at all.

Similarly, if the rule makes a reference to the advice being given to `Client` and involves a relation that restricts it to be financial in nature, the match of `isAdvisorOf` would be constrained to be with a subrelation of `isFinancialAdvisorOf`; i.e., the match would be improved. This would improve the other matches too, by constraining `Broker` to have financial accreditation and the right to work in a particular market.

- **Capturing implicit restrictions:**

  Reasoning about the rules in a KBS can also yield constraints on potential matches that are not obvious from the (local) schemas to be integrated, but which are nevertheless important in the global schema. These concern features that are implicit in the original local schemas, but need to be considered explicitly in the global schema, because it might contain entities that do not have them. Note that the global schema must eventually contain some class that has the relevant features. This is because if the global schema does not initially subsume a given local schema, it is extended until it does.

  Consider a KBS with a rule that selects hotels for handicapped persons by looking at the preferences of a given person and the features available in a given hotel. Assume that the KBS implicitly considers only hotels that are wheelchair accessible, because those are the only ones listed in its local knowledge base. Suppose this KBS is to be integrated with a general database containing entries for hotels in general. For this integration to be correct, the generated articulation axioms must relate `Hotel` in the local schema to `Hotel` $\cap$ `Accessible` in the global schema.

  When `Person` in the original KBS is matched with `HandicappedPerson` in the global schema, the relation `AcceptableHotel` is constrained to match with the subrelation of `AcceptableHotel` in the global schema that applies to the given subclass of `Person`. This forces `Hotel` to be matched with `AccessibleHotel`, as desired. As a result, the correct articulation axiom would be generated, so that when the KBS queried its local knowledge base for hotels, it would now query the integrated database for hotels that are wheelchair accessible. The rule can then fire as before.

The above heuristics only help us come up with plausible matches between concepts in the local schema and those in the global schema.

Constructing Articulation Axioms

An articulation axiom is then constructed for each match found. For example, the match between the relational attribute `phone` and the Cyc slot `phoneNumber` yields the axiom

$$ist(Cyc\ phoneNumber(?L\ ?N)) \Longleftrightarrow ist(AAA\ phone(?L\ ?N))$$

which means that the `phone` attribute definition determines the `phoneNumber` slot in the global schema, and vice versa. Articulation axioms are generated automatically by instantiating stored templates with the matches found.

Discussion

We described an ongoing experiment in integrating information models. In our approach, the integration of resource schemas is based on articulation axioms defined between two contexts: the context of a resource schema and a global schema context provided by the Cyc knowledge base. Our methodology is based on the following principles:

- Existing data should not have to migrate or be modified to achieve integration.
- Existing applications should not have to be modified due to integration.
- Users should not have to adopt a new language for communicating with the resultant integrated system, unless they are accessing new types of information.
- Resources should be able to be integrated independently, and the mappings between the schemas of different resources that result fro this integration should not have to change when additional resources are integrated.

We satisfy this last principle by using an existing global schema to resolve inconsistencies. Information resources are related to this global schema—not directly to each other—with the relationships expressed in terms of mappings between each individual schema and the global schema. As a result, the relationships can be constructed independently; they do not have to be altered when other resources are integrated in the future; and we need only N sets of mapping functions for n resources, instead of $n(n \Leftrightarrow 1)$ functions in an approach where there is no common model and language. Another advantage is that an update of a local schema is propagated only to the global schema, and only one mapping function has to be recalculated. We believe that this calculation would typically involve only a few rules of the mapping.

The above principles are incorporated in an integration tool for assisting an administrator in integrating a resource and a transaction tool for providing users and applications with access to the integrated resources. The integration tool uses an extensive set of semantic properties to represent an information resource declaratively within the global schema and to construct bidirectional mappings between the resource and the global schema. The mappings are used by the transaction tool to translate queries and updates written against any local schema into the appropriate form for each information resource. These tools constitute a part of the semantic services of Carnot [Cannata, 1991], under development at MCC. Carnot will enable development of open applications that can be tightly integrated with information stored on existing closed systems. The semantic services layer of Carnot provides facilities to specify and maintain the semantics of an organization's integrated information resources.

Thus our approach to resource integration allows a user or application to use a familiar local schema, while still benefiting from newly added resources. While it is not meant to replace a human designer, it does help significantly by presenting the designer with plausible matches from which to choose—a task that can be quite demanding for an unassisted human when the schemas involved are large and complex.

References

M. Ahlsen and P. Johannesson, "Contracts in Database Federations," in S. M. Deen, ed., *Cooperating Knowledge Based Systems 1990*, Springer-Verlag, London, 1991, pp. 293–310.

O. P. Buneman, S. B. Davidson, and A. Watters, "Querying Independent Databases," *Information Sciences*, Vol. 52, Dec. 1990, pp. 1–34.

P. E. Cannata, "The Irresistible Move towards Interoperable Database Systems," *First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 7-9, 1991.

C. Collet, M. N. Huhns, and W.-M. Shen, "Resource integration using a large knowledge base in Carnot," *IEEE Computer*, Vol. 24, No. 12, Dec. 1991, pp. 55–62.

R. V. Guha, "Micro-theories and Contexts in Cyc Part I: Basic Issues," MCC Technical Report Number ACT-CYC-129-90, Microelectronics and Computer Technology Corporation, Austin, TX, June 1990.

D. Heimbigner and D. McLeod, "A Federated Architecture for Information Management," *ACM Transactions on Office Information Systems*, Vol. 3, No. 3, July 1985, pp. 253–278.

D. Lenat and R. V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 267–296.

S. B. Navathe, S. K. Gala, S. Geum, A. K. Kamath, A. Krishnaswamy, A. Savasere, and W. K. Whang, "Federated Architecture for Heterogeneous Information Systems," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

A. P. Sheth and S. K. Gala, "Attribute Relationships: An Impediment in Automating Schema Integration," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 183–236.

J. de Souza, "SIS—A Schema Integration System," *Proc. Fifth British National Conference on Databases (BNCOD5)*, Cambridge University Press, 1986, pp. 167–185.

R. Wang and S. E. Madnick, "Facilitating Connectivity in Composite Information Systems," *Data Base*, Fall 1989, pp. 38–46.