Dynamic Integration in Multidatabase Systems[1]

Wen-Syan Li
CIMIC, Rutgers University

Chris Clifton[2]
The MITRE Corporation

The first step in interoperating among multidatabases is semantic integration: Producing attribute correspondences that describe relationships between attributes or classes in different database schemas. Dynamic integration requires the ability to automatically extract database semantics, express them as metadata, and match semantically equivalent data elements to produce attribute correspondences. This process can not be "pre-programmed" since the information to be accessed is heterogeneous. In this paper we present an architecture supporting dynamic integration. We first overview a tool, Semint, for automated semantic integration that helps database administrators generate attribute correspondences. We then introduce a novel framework for dynamic integration and a query language for multidatabase systems that uses Semint as part of a complete semantic integration service. Our framework supports dynamic integration as well as incremental integration. We show the advantages of our framework in an environment where full integration is not desired or complete knowledge of the databases to be integrated is unavailable.

Applications in a wide variety of industries require access to multiple heterogeneous databases due to company mergers, the introduction of new database technology, or integrating information across departments. There are two ways to integrate existing information systems: reengineering and interoperation. In reengineering, the application logic, data definition, and data from the old systems are transferred into a new system. The advantage of reengineering is that the new systems are easier to maintain. However, this process is expensive and complicated.

The second option is to allow the existing systems to interoperate as a multidatabase system by putting a new, standard interface on existing databases. This preserves the data and applications from the existing databases, yet allows access to the data from new multidatabase applications. This transforms the existing databases into open databases. Interoperability allows computing resources to be shared, thereby giving organizations a new resource and a uniform, enterprise-wide and database-independent view of data.

The goal of integrating interoperable multidatabases is to provide an enterprise-wide information system where users can access information through an unified interface without knowing the details of each participating component database. This gives a computer paradigm of a large number of information sources that are heterogeneous in semantics and format. Automatically extracting and understanding the semantics of information and format conversion are important issues to information integration and such metadata should be based on the contents of the information available.

Techniques essential to information integration include extracting semantics, transforming formats, identifying attribute correspondence, resolving heterogeneity, multidatabase query processing, and data integration. In order to answer queries in multidatabase systems, three distinct processes need to be performed by the user, database administrator, and/or system as shown in Figure 1. The Schema Integration process includes a possible schema transformation step, followed by correspondence identification, and an object integration and mapping construction step [Parent and Spaccapietra, 1995]. In Query Processing, global queries are reformulated into subqueries, the subqueries are executed at the local sites,

Manuscript originally submitted September 1, 1995; Revised September 25, 1995; Accepted October 4, 1995 for publication.

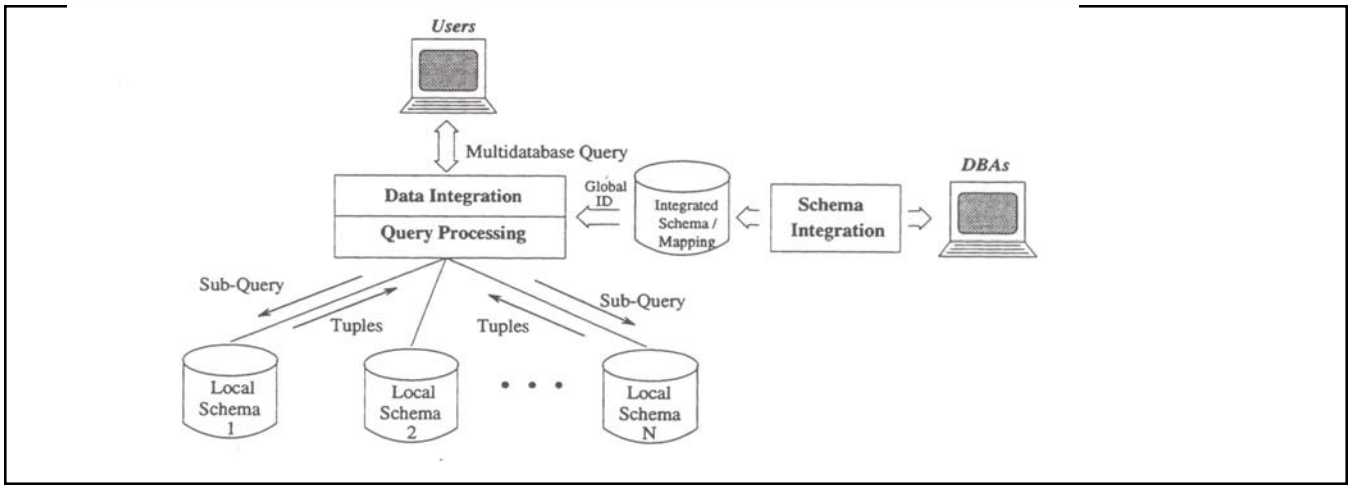

Figure 1: Multidatabase Query Processing

and their results are assembled at a final site. The Data Integration process is complimentary to Query Processing, i.e., it determines how the results from different local databases should be merged and presented at the final site.

The concept of dynamic integration rests on the premise that knowledge (metadata) needed to integrate information is provided by or can be extracted from databases directly. This allows computer "mediators" to handle the heterogeneity so end-users and non-computer experts can access useful data. The formats of data and its semantics presentation are not expected to be standardized, developing schemas to integrate this data is a mediation task [Wiederhold, 1992]. An automatic process for extracting and transforming data semantics is also an important issue as the volume and variation of accessible data increases.

The Problem

We are concerned with one area in interoperable multidatabases: dynamic integration. We argue that attribute correspondence identification is a primary bottleneck in integration and query processing of multidatabases. For example, users of heterogeneous databases may issue a query that joins two relations in different databases; to do this they must know what attributes in the relations can be used as a join key. If the attribute correspondences between the databases are known, finding a join key is a simple process. However, if these databases have not been integrated or users are not familiar with the attribute correspondences, multidatabase queries cannot be issued. Attribute correspondences are also essential to data integration —merging results from component databases.

We argue that the solution to this problem is to automate the process of attribute correspondence identification. Human effort is still necessary in database integration unless the semantics of data can be captured completely and the techniques of artificial intelligence is mature to act as an domain expert. Domain knowledge is a necessary part of any semantic integration system; we cannot count on the available metadata completely and accurately describing the semantics of the data. We categorize the approaches to attribute correspondence identification, based on when human effort is required, as follows:

Manual integration: Human effort is needed to collect metadata from databases that captures the semantics of data. The semantics may be embodied within a database model, a conceptual schema, application programs, or the minds of users. Humans need to be involved in collecting metadata and matching corresponding attributes. The manual integration is tedious and time consuming. Some corporate experiences of manual integration were described in [Ventrone and Heiler, 1994].

Semiautomated integration: In this approach, tools are used to collect metadata and generate candidates for corresponding attributes. DBAs (Database Administrators) need to be involved in checking and confirming the recommendations of the tools. Multidatabase queries can not be issued until attribute correspondences are generated. An example of this kind of tool is described in [Li and Clifton, 1994,Li and Clifton, 1995].

Dynamic integration: Users can issue queries before attribute correspondences are generated. The attribute correspondence identification process is carried out dynamically. Collecting metadata and generating candidate attribute correspondences are automated. Users need to check multidatabase query results rather than attribute correspondences. As users are generally "domain experts" on data stored in databases (or at least the data they are interested in), as opposed to experts in schema design, the query results are more meaningful to users than attribute correspondences.

An interface providing semiautomated or dynamic integration needs to utilize available metadata that can be automatically extracted from databases. An automated tool for identifying attribute correspondences can help to open this bottleneck, allowing dynamic integration and query process-

ing in multidatabases.

Our dynamic semantic integration procedure makes use of an automated tool based on neural networks to identify candidate attribute correspondences. The query processing step uses these attribute correspondences to reformulate multidatabase queries.

Paper Organization

The rest of this paper is organized as follows. In Section 2 we review issues and related work in multidatabase and federated database systems. In Section 3 we briefly describe the semantic integration tool, Semint, used in our dynamic integration framework. Section 4 describes how Semint can assist DBAs in generating attribute correspondences to develop an integrated schema (semiautomated integration). In Section 5 we provide a new framework for multidatabase query processing, dynamic integration, that does not required advance determination of attribute correspondences. We provide sample queries to illustrate our method and discuss the advantages of our approach. Finally, we conclude this paper with a discussion of the benefits of this method and areas for future work.

Related Work

We will first review some general work in multidatabase systems, then describe previous work in data and semantic integration. We then discuss how this work differs from previous systems, and the added benefit provided.

Multidatabase Systems

Early work in multidatabase architectures focused on procedures to merge individual schemas into a single global conceptual schema. [Batini et al., 1986] surveyed 12 methods for constructing global schemas. The global schema multidatabase approach requires complete integration: the global schema must map all local schemas to a single global view. The amount of knowledge required about local schemas and how to identify and resolve heterogeneity among the local schemas is a major problem with this approach. The global schema must be developed before any queries can be issued. Changes to local schemas must be reflected by corresponding changes in the global schema. This causes major difficulties in maintaining the global schema. Because of the complexity of a global schema, a small change to a local schema (e.g. add or delete an attribute) may require huge changes to the global schema. To visualize this, assume 50 local schemas have been merged into a global schema. If each local schema changes once per year, the global schema DBA needs to change the global schema weekly. Goh et al. (1994) argues existing *a priori* or static integration strategies might provide satisfactory support for small or static systems, but not for large-scale interoperable database systems operating in a dynamic environment.

Federated databases [McLeod and Heimbigner, 1980,Parent and Spaccapietra, 1995] are an approach that resolves some of the problems associated with global schema. Federated databases only require partial integration. A federated database integrates a collection of loosely coupled local database systems by supporting interoperability between pairs or collections of the local databases rather than through a complete global schema. However, although the impact of a change to a local schema may be smaller; any change to the local schema may require some change to the federated schema. Maintaining these mappings is still difficult.

Multidatabase languages are an attempt to resolve some of the problems associated with a global schema. With these systems, no global schema (not even a partial one) is maintained. Examples of multidatabase languages include [Hwang et al., 1984,Litwin et al., 1989,Scheuermann and Chong, 1994,Bright et al., 1994]. This approach puts the integration responsibility on users by providing functionality beyond standard SQL to allow users to specify integration information as part of the query.

This is a heavy burden to place on users, however. We instead view the multidatabase language as an intermediate language, to be used for query processing after schema integration issues have been resolved. We use MSQL [Litwin et al., 1989] as the basis for our query language.

Data Integration

Data integration is concerned with difficulties of combining data values that reflect the same information for the same entity from multiple databases. Multidatabase languages are usually similar to SQL in their standard capabilities, but provide a global name space to hide the heterogeneity of accessing particular remote databases (database instances) and special functions to ease problems of data heterogeneity such as new types of information presentation. These functions are provided for data integration. Structural differences are usually resolved by one of three methods: outerjoin operations (Rosenthal and Reiner, 1984), generalizations (Hwang et al., 1984), or multiple relations (Litwin et al., 1989). However, these methods still require static data integration; how to resolve structural differences must be known before the query is issued.

Schema Integration

Parent and Spaccapietra (1995) identified three major steps in the database integration process: Pre-integration to rearrange input schemas to be more homogeneous, correspondence identification, and Integration. Multidatabase languages eliminate the need for complete integration, but correspondence identification is still required to determine the Global ID. Sheth and Kashyap (1992) argued that identifying semantically related objects and then resolving the schematic differences is the fundamental question in any approach to database system interoperability.

We refer to the process of identifying corresponding attributes as semantic integration. In semantic integration, attributes (classes of data items) are compared in a pairwise fashion to determine their equivalence. Li and Clifton (1994) categorized metadata that can be automatically extracted from databases as: attribute names (the dictionary level), schema information (the field specification level), and data contents and statistics (the data content level). Problems encountered here are that synonyms occur when objects with different names represent the same concepts, and homonyms occur when the names are the same but different concepts are represented. From GM's efforts in integration [Premerlani and Blaha, 1994], attribute names were not sufficient for semantic integration; only a few obvious matches were found. However, similarities in schema information were found to be useful. For example, it was discovered in one case that attributes of type char(14) were equivalent to those of char(15) (with an appended blank).

Sheth et al (1988) concentrated on interactive support for manual semantic integration. MUVIS [Hayne and Ram, 1990] is a system for view integration where object equivalence is determined by using "fixed rules" to compare the aspects of each and computing a weighted value for similarity and dissimilarity. [Li and Clifton, 1994] applied neural network techniques to semantic integration, the knowledge of how to match equivalent data elements is "discovered" directly from metadata, not "pre-programmed". A more complete overview of Semint is given in Section 3. Some of these approaches are highly automated, but the integration must still be done before the query is written.

This paper presents a technique where this integration is dynamic; any manual effort in schema integration is done only after the query is complete, and need only be done for the results of the query. Therefore the only "static" information needed is how to query different DBMS types (gateways and standards are likely to solve this problem). All database specific integration is handled after the query is issued. This reduces the *a priori* or static effort required for schema integration to that of multidatabase languages (query translation to different DBMS types), while relieving the user of providing "federated schema" knowledge when issuing the query. We will now give more detail on Semint, then discuss our dynamic integration model.

Semantic Integration Process

Semint (SEMantic INTegrator) [Li and Clifton, 1995] is a system for semantic integration based on [Li and Clifton, 1994]. Semint makes use of the fact that attributes in different databases that represent the same real-world concept will likely have similarities in schema designs, constraints, and data value patterns. For example, employee salaries in two databases will probably be numbers greater than 0 (this is structural similarity, as it can be determined from constraints). The same can be said for daily gross receipts. However, the range and distribution of data values will be very different for salaries and gross receipts. From this we can determine that two salary fields probably represent the same real-world concept, but gross receipts are something different.

Databases to be integrated are accessed directly using automatic "catalog parsers". Figure 2 outlines the semantic integration procedure in Semint. In this procedure, DBMS specific parsers extract metadata (schema and data content statistics) from databases and transform them into a single format (so they can be compared). Then, a classifier is used to learn how to discriminate among attributes in a single database. The classifier output is used to train a neural network to recognize categories; this trained network can then determine similar attributes between databases.

Note the only human input is to specify DBMS types and database connection information and to examine and confirm the output results (similar pairs of attributes and the similarity between them). Other processes are fully automated. Semint automatically extracts schema information from a database, analyzes data contents to generate statistics, transforms database metadata into a single format, builds and trains neural networks, and uses trained neural networks to determine their similarity and identify similar attributes (whose degrees of similarity are greater than similarity threshold set by the users). Semint has three components: A DBMS specific parser, Classifier, and Neural Network.

DBMS Specific Parser

The information used in Semint is extracted from the database automatically. The parser queries the data dictionaries to determine what is in the database. Based on this, Semint generates queries to extract information about each attribute.

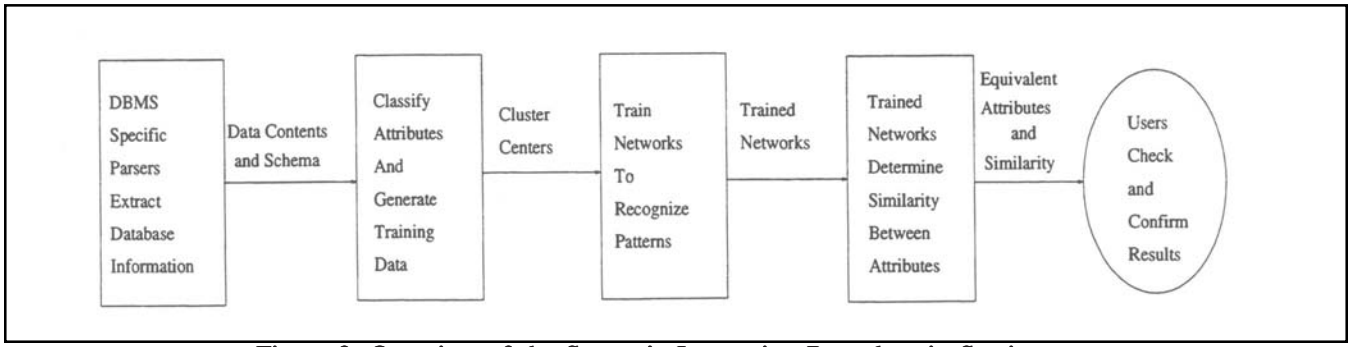

Figure 2: Overview of the Semantic Integration Procedure in Semint

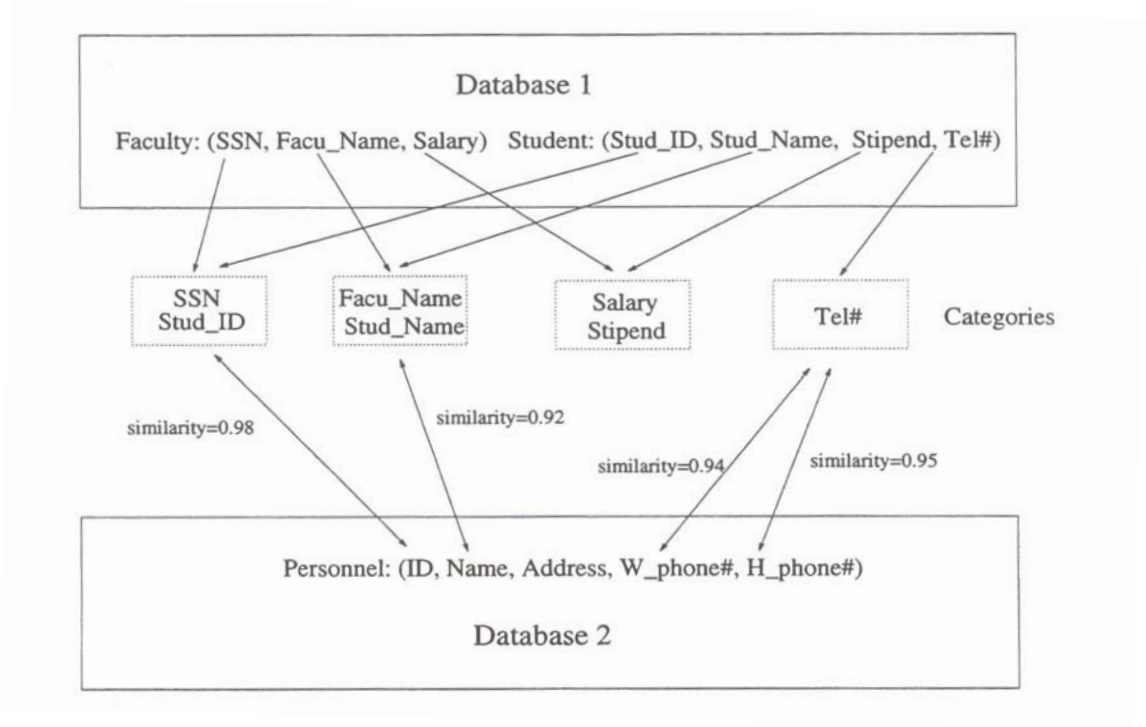

Figure 3: Example of Attribute Correspondence Determination Process

Although different DBMSs use different data dictionaries to contain schema information and integrity constraints, these DBMS specific parsers are similar. Semint automatically extracts schema information and constraints from the database catalogs and statistics on the data contents using queries over the data. The schema information used in Semint includes data types, length, scale, precision, and existence of constraints (primary keys, foreign keys, candidate keys, value and range constraints, disallowing null values, and access restrictions). The statistics on data contents used in Semint include maximum, minimum, average (mean), variance, coefficient of variance, existence of null values, and existence of decimals. The information extracted from different databases is then transformed into a single format and normalized. The parser output is a set of vectors; each vector presents the characteristics (schema and data contents) of an attribute.

Classifier

The available information from an individual database discussed above is used as input data for a classifier to categorize attributes within a single DBMS. Semint uses the Self-Organizing Map algorithm, an unsupervised learning algorithm, as the classifier. We have adapted this algorithm so that users can determine how fine these categories are by setting the radius of clusters (threshold) rather than the number of categories; if desired users may examine the output and adjust this threshold to cluster like attributes together. The output of the classifier is the vectors of cluster center weights.

Neural Networks

The output of the classifier is then used as training data for a back-propagation network, a supervised learning algorithm. The "supervision" is that target results are provided; however as these target results are the output of the classifier, no user supervision is needed. We use this to train a network to recognize input patterns and give degrees of similarity. The similarities are a measure of how close the vector describing an input pattern is to each of the vectors of the training data. The "distance function" for close is not predefined, but is learned directly from the database semantics during the training process, and will vary depending on the information contained in the database (allowing Semint to adjust itself to different database domains). Therefore similarity does not correspond to a percentage or fixed distance function, but is a domain-specific value that can be used to rank the likelihood that two attributes reflect the same real-world information.

To determine similar attributes between two databases, users take the network trained for one database, and use information extracted from the other database as input to this network. The network then gives the similarity between each pair of attributes in the two databases. System users check and confirm the output results.

In Figure 3, the attributes in database 1 are clustered into four categories using classifier and the cluster center weights are then used to train neural network. After the neural network is trained (it knows the "signatures" of these four categories), it matches the attributes in database 2 with these categories classified from database 1.

In the example shown in Figure 4, we want to integrate Faculty and Student databases. Semint first uses DBMS spe-

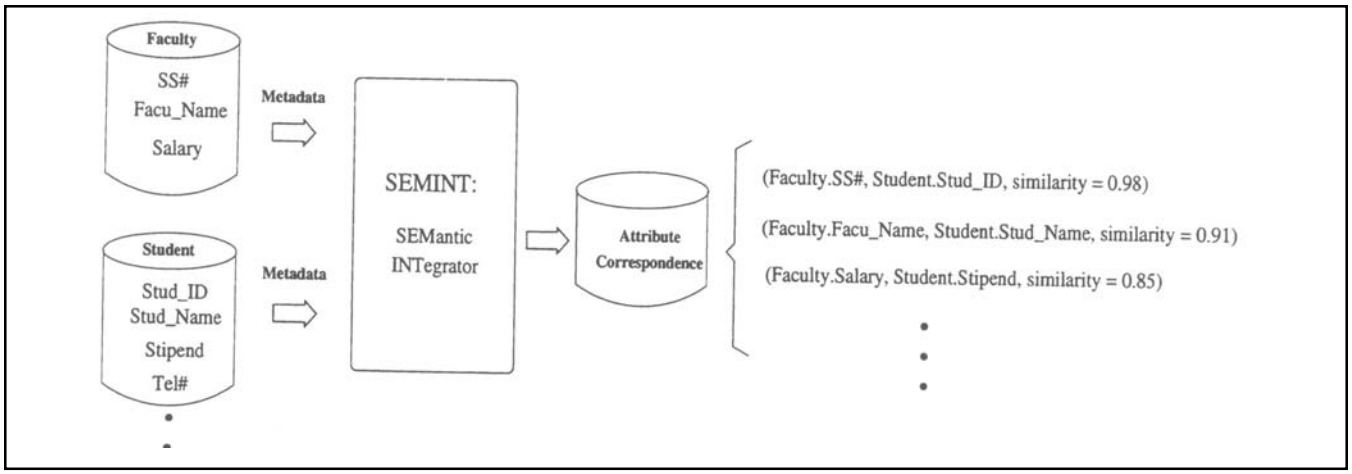

Figure 4: Example of Semantic Integration in Semint

cific parsers to extract metadata (schema design, constraints, and data content statistics) from these two databases. The metadata form "signatures" of the attributes in the Faculty and Student databases. These are used as training data for neural networks to recognize these attributes' patterns. The trained neural network can then generate attribute correspondences between the Faculty and Student databases and the degree of similarity of each pair of corresponding attributes as shown on the right side of Figure 4. How to match corresponding attributes and determine their similarity is "learned" during the training process directly from the metadata used as the training data.

The semantic integration process can be static (done in advance) or dynamic (carried out on demand when required by a query) as described in Section 5. In either case, the neural network training process needs to be done only once per database. Once the network is trained, integration with a new database only requires running information from the new database through the neural network (a sub-second process [Li and Clifton, 1994]). This automated semantic integration ability of Semint makes it a useful tool for dynamic integration.

Semiautomated Integration in Multidatabases

One problem in multidatabase system integration is that DBAs are responsible for collecting metadata from component databases and determining matching attributes. It is usually done manually; as a result, it is tedious and inefficient. In this section, we show how Semint is used to assist DBAs in performing this mapping and constructing a conceptual integrated schema. We use an example scenario to demonstrate this process.

Constructing the Conceptual Integrated Schema

Assume that our multidatabase integrates two local schemas, namely FACULTY (SS#, Faculty_Name, Salary) and STUDENT(Stud_ID, Stud_Name, Stipend, Tel#), as shown in Figure 5. The DBAs first specify DBMS type and connection information. Semint then accesses the target databases to generate attribute correspondences. The attribute correspondences and their similarity as recommended by Semint (with similarity threshold set to 0.8) are as follows:

(Faculty.SS#, Student.Stud_ID, similarity = 0.98)
(Faculty.Facu_Name, Student.Stud_Name, similarity = 0.91)
(Faculty.Salary, Student.Stipend, similarity = 0.85)

Now DBAs use the attribute correspondences recommended by Semint as a template to construct a conceptual integrated schema. The conceptual integrated schema, Employee, is constructed following the procedure below:

Step 1: DBAs check and confirm the result recommended by Semint.

Step 2: DBAs assign a new attribute name for each pair of corresponding attributes to resolve naming conflicts. After this step, DBAs generate a table such as:

Employee.ID = (Faculty.SS#, Student.Stud_ID)
Employee.Name = (Faculty.Facu_Name, Student.Stud_Name)
Employee.Salary = (Faculty.Salary, Student.Stipend)

Step 3: DBAs then add attributes that are found in only one database, such as Student.Tel#. Student.Tel#.

The conceptual integrated schema is now completed:

[Conceptual Integrated Schema]	[Component Database Schema]	
Employee	**Faculty**	**Student**
ID	SS#	Stud_ID
Name	Facu_Name	Stud_Name
Salary	Salary	Stipend
Tel_num		Tel#

[Faculty]

SS#	Facu_Name	Salary
493-45-8735	John	$100,000
956-45-0456	Robert	$80,000
849-45-0500	Patricia	$60,000
485-95-6784	Larry	$20,000

[Student]

Stud_ID	Stud_Name	Stipend	Tel#
476-34-5748	Jason	$11,000	674-7456
958-46-3256	Steven	$0	765-0945
485-75-2374	Patricia	$0	767-5134
485-95-6784	Larry	$12,000	767-0900

Figure 5: Faculty and Student Databases

Query Language and Processing

Users can now submit a multidatabase query against Employee directly without specifying intradatabase join conditions.[3] The multidatabase query extends the standard SQL by allowing users to specify attributes in a conceptual integrated schema such as Employee.ID. A brief description of the extended multidatabase query language is:

MULTISELECT attribute names in the conceptual integrated schema
FROM conceptual integrated schema
WHERE selection conditions

The "MULTISELECT" clause specifies which attributes to retrieve. The "FROM" clause specifies the names of conceptual integrated schema. The "WHERE" clause specifies the selection conditions. We now use a sample query to demonstrate multidatabase query processing. We know that Employee contains both faculty and student information and we know some students may not have a stipend. Imagine that we are preparing a tax report. We want to know

Who is on the university payroll?

The query can be posed as:

MULTISELECT *
FROM Employee
WHERE Employee.Salary > 0

This query is then reformulated into two subqueries for databases Faculty and Student and executed. We have shown how Semint assists in generating attribute correspondences, that are then used as a template to construct a conceptual integration schema. Although some tools (such as Semint described here) can assist DBAs in collecting metadata and matching related attributes, attribute correspondences and a conceptual integration schema still need to be generated before multidatabase queries can be issued. This poses a problem when the need for integration is not known in advance (for example, browsing for information in various databases). It is also requires that complete knowledge of the databases to be integrated be available (or learned by the DBA), even if complete knowledge would not be required to answer any particular query.

In the next section, we present a dynamic integration framework to solve the problems described above. In this framework, Semint is used as a tool to generate attribute correspondences dynamically, a conceptual integration schema need not be generated before multidatabase queries can be issued.

Dyamic Integration in Multidatabases

Dynamic data integration methods, such as MSQL, require that mappings between attributes be known to generate a query. A complete Conceptual Integrated Schema is not required, however the attribute correspondences must be known for any "global" attributes used in the query. This requires some schema integration (attribute correspondence identification). We present an approach for dynamic data integration and query processing where we release the assumption that attribute correspondences are known in advance. MSQL is extended to allow the absence of information about the attribute correspondences, and Semint is used as a dynamic integration tool. In our framework, users specify wildcards for attribute names where the mapping is unknown or schema integration is not available. Semint determines potential correspondences to replace the wildcards. The multidatabase query (or queries if more than one candidate arises for each correspondence) are then executed as regular MSQL queries. The result is presented to the user as a set of tables, ranked by the degree of attribute correspondence used for the mapping used to generate the query producing that table.

This reduces the *a priori* or static effort required for schema integration to that of multidatabase languages (query translation to different DBMS types), while relieving the user of providing "federated schema" knowledge when issuing the query. Semantic integration is handled by presenting a query result for each likely attribute correspondence. Thus the user need only determine which of the possible results makes sense, as opposed to determining the attribute correspondence in advance.

We see three scenarios for dynamic integration, depending on the level of knowledge of attribute correspondence that

a user has when issuing a query:

1. Two databases are well understood: the attribute correspondence is known;
2. Only one database is well understood: the desired attribute is known for one database, however the corresponding attributes in other databases needs to be determined; and
3. Neither database is well understood; finding the appropriate mappings needs to be done "from scratch". This case only applies when either:
   - The unknown quantity is a join condition for the databases; or
   - The goal is to select all attributes from both databases.

In this paper, we present only the case where we are looking for a join criteria (Global ID). This covers all three scenarios. We briefly discuss other cases, such as aggregation (e.g., combining salary and stipend) at the end of this section.

Query Language

Our approach works in all three scenarios, so users can submit a multidatabase query whether or not a Global ID is known. This requires extensions to the MSQL to allow specifying a Global ID, or requesting that one be determined automatically. The extensions are shown in Figure 6. The "MULTISELECT" clause specifies which attributes to retrieve. The "WHERE" clause specifies what is known about Global IDs (GIDs), and what needs to be determined. If the GID is known in advance (the first scenario), all attributes are specified.

```
MULTISELECT  attribute names
FROM  relation1 relation2 ... relationn
WHERE   relation1.( attribute | *) = relation2.( attribute |  *)  = ...  =
      relationn.( attribute |  *) [  WITH SIMILARITY >   threshold ]
                    [  BEST   max_matches ]
```

Figure 6: Multidatabase Query Language With Dynamic Integration

Data integration and query processing when the GID is known are as described in Litwin et al. (1989). In the second scenario (one attributes is specified as "*") and third scenario (both attributes are specified as "*"), Semantic Integration needs to be carried out to generate attribute correspondences to be used as candidate global IDs. The clauses "WITH SIMILARITY>threshold" and "BEST max_matches" are optional, and are used to restrict the number of potential GID candidates to those where Semint gives a similarity measure greater than threshold, and if a large number meet that criteria, only the best max_matches candidates (if these are not specified, defaults are used).

The overview of our framework architecture is shown in Figure 7. If no global ID is provided, the multidatabase query parser first uses the Semantic Integration process (Semint) to attempt to find attribute correspondences (as shown in Figure 9). Semint accesses the local databases to retrieve metadata describing the database attributes. This metadata is used to generate attribute correspondences: A list of semantic related attributes and their similarity (between 0 and 1). Assume that our multidatabase integrates two local schemas, namely FACULTY (SS#, Faculty_Name, Salary) and STUDENT(Stud_ID, Stud_Name, Stipend, Tel#), as shown in Figure 8. The attribute correspondences and their similarity generated by Semint from Faculty and Student (as shown in Figure 8) with similarity threshold set to 0.8 are as follows:

(Faculty.SS#, Student.Stud_ID, similarity = 0.98)
(Faculty.Facu_Name, Student.Stud_Name, similarity = 0.91)
(Faculty.Salary, Student.Stipend, similarity = 0.85)

These corresponding attributes (the pairs of attributes with high similarity) are used as candidate global IDs.

The candidate global ID's are selected from this list. If

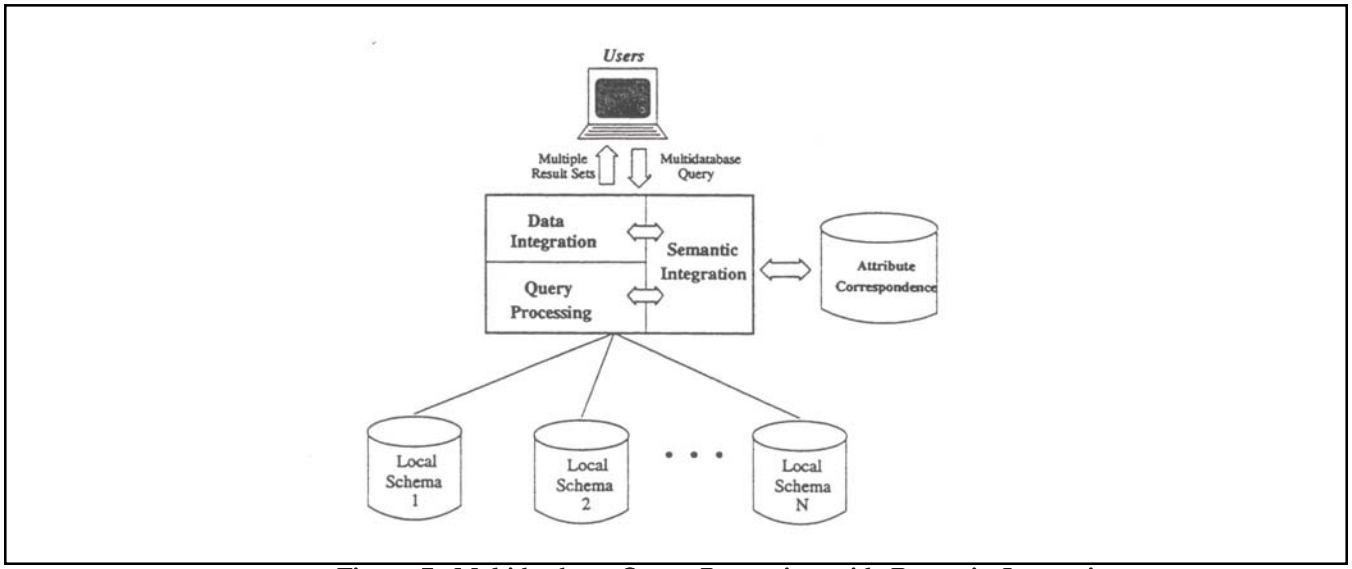

Figure 7: Multidatabase Query Processing with Dynamic Integration

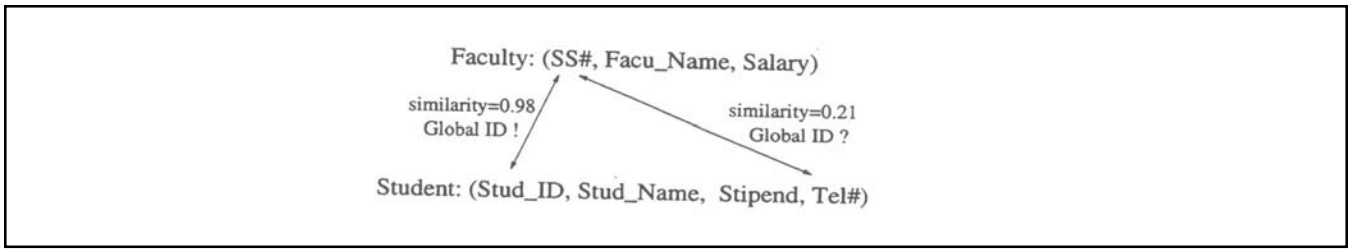

Figure 8: Possible Global IDs in Faculty and Student

the user specifies one of the local IDs, any attributes that match the given ID with similarity higher than the given threshold (or a default) are used. If no local ID is given, any pair of attributes with high enough similarity are used. This results in a collection of "scenario 1" MSQL queries, each with a different Global ID.

Attribute correspondences are generated dynamically before the actual multidatabase query is processed. This is done on a demand basis: If the global ID is known (or has previously been determined and saved in a list of attribute correspondences) no semantic integration is needed. If the global ID is unknown, Semint is run to find candidates. [4]

The *Query Processing* step (Figure 10) decomposes the global MSQL query (or queries if there are multiple candidate GIDs) into component database subqueries and submits them to the local databases. The subqueries are executed at the local sites and tuples are returned to the global site.

The *Data Integration* step receives these returned tuples and merges them as multiple sets of answers ranked by the similarities in the attribute correspondence list. A set of answers is generated for each of the candidate GIDs. Each represents a possible answer to the query; the correct one depends on the correct GID. These are ranked based on the "confidence" in the correctness of the result. The degree of confidence of a result table is based on the similarity of attribute correspondence used as a GID. For example, the degree of confidence of table using GID (Faculty.SS#,Student.Stud_ID) is 0.98 since the similarity between Faculty.SS# and Student.Stud_ID is 0.98.

Dynamic Integration and Query Processing Procedures

The procedure of dynamic integration query processing with ranked answer sets is outlined below:

Pre-multidatabase-query process (semantic integration, Figure 9)

Step 1: The users submit a multidatabase query to retrieve semantically similar data items. The "WHERE" clause specifies the type of global ID assumption.

Step 2: If the global ID is unknown, Semantic Integration Process (Semint) at the global site extracts metadata from the local databases.

Step 3: Semint uses the metadata extracted in Step 2 to generate attribute correspondences as candidate GIDs according to the (user-specified or default) similarity threshold.[5]

Multidatabase Query Processing (Figure 10)

Step 4: Multidatabase Query Processing re-formulates the original multidatabase query into multiple multidatabase queries according to attribute correspondences. One

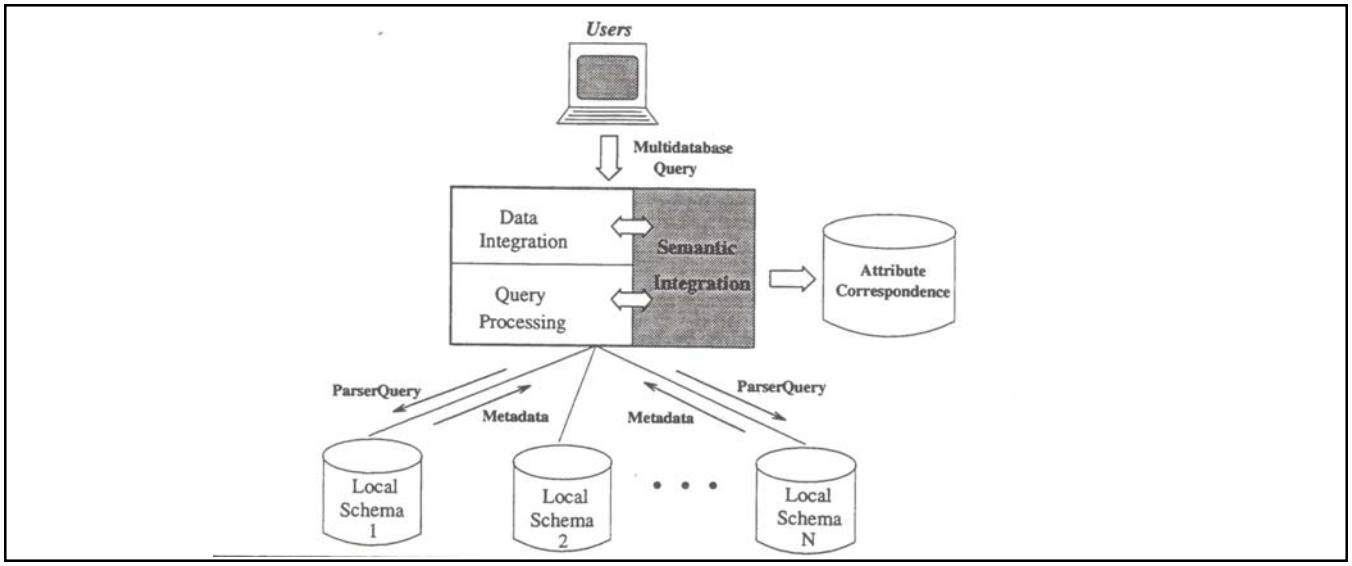

Figure 9: Pre-Multidatabase-Query Processing: Semantic Integration

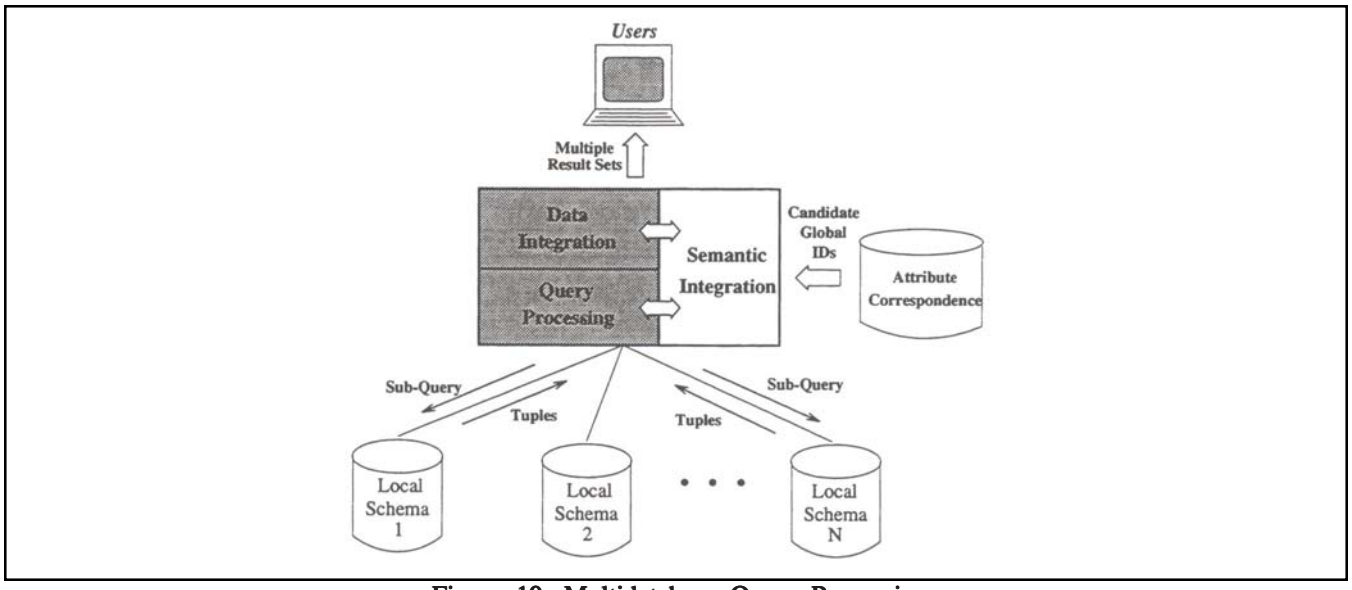

Figure 10: Multidatabase Query Processing

multidatabase query is generated for each candidate GID from the attribute correspondences list. [6]

Step 5: The multidatabase query pre-compiler generates subqueries for each multidatabase query generated in Step 4 and then submits subqueries to the local databases.

Step 6: The local databases return the result tuples of the subqueries to the originating site.

Step 7: The Data Integration process merges the intermediate results from various sites by consulting the attribute correspondence list. The results are presented to the users as tables with degrees of confidence (ranked result sets). One set of results is generated for each pair of corresponding attributes.

Note: Steps 2 and 3 only need to be done once, and can be done in anticipation of possible queries to improve query response time. Extending Semint to identify "compound keys" and multiple query optimization issues are left for future research.

Example Scenario

In this section, we use some sample queries to demonstrate the dynamic integration and query processing of our approach. Imagine that we are planning a university budget. We want to know:

What are the salaries of student instructors?

The salary of a student instructor may come from two sources: Faculty salary from the University and student stipend from the Graduate School. The faculty salary information is stored in the Faculty database and student stipend information is stored in the Student database, as shown in Figure 11. Here we only list relevant attributes for ease of illustration.

As we discussed in Section 5, we see three scenarios based on database knowledge: Two databases are well understood so that a global ID is known, only one database is well understood (a local ID is known, but the corresponding ID needs to be determined by semantic integration), and neither database is well understood (global IDs need to be determined by semantic integration).

Global ID is Known. In the first scenario, the user knows that Faculty.SS# and Student.Stud_ID form a valid global identifier. The query can be posed as:

```
MULTISELECT *
FROM Faculty, Student
WHERE (Faculty.SS# = Student.Stud_ID)
```

The result of this query is shown in Figure 12. In this case, the semantic integration is static (done in advance) so the GID is known.

ID is known for Faculty Database. In this section we discuss how our approach works in the second scenario: Only one database is well understood. A local ID is known for this

[Faculty]			[Student]			
SS#	**Facu_Name**	**Salary**	**Stud_ID**	**Stud_Name**	**Stipend**	**Tel#**
493-45-8735	John	$100,000	476-34-5748	Jason	$11,000	674-7456
956-45-0456	Robert	$80,000	958-46-3256	Steven	$0	765-0945
849-45-0500	Patricia	$60,000	485-75-2374	Patricia	$0	767-5134
485-95-6784	Larry	$20,000	485-95-6784	Larry	$12,000	767-0900

Figure 11: Faculty and Student Databases

(SS#=Stud_ID)	Facu_Name	Salary	Stud_Name	Stipend	Tel#
485-95-6784	Larry	$20,000	Larry	$12,000	767-0900

Figure 12: Result using (SS#,Stud_ID) as GID with Degree of Confidence = 0.98

database; however, the global ID needs to be determined by semantic integration. We know we need to access the Faculty and Student databases. We are familiar with Faculty; however, we have little knowledge about Student. We know the two databases should contain some similar data items such as salary, social security number, and name. We can submit the follow multidatabase query to retrieve the salaries of student instructors using Faculty.SS# as part of the global ID. Because the Student database is not well understood, we specify the corresponding attribute in Student as "*". Semint will then determine the possible corresponding attributes in Student to use with Faculty.SS# as the GID. The query follows:

```
MULTISELECT *
FROM Faculty, Student
WHERE  Faculty.SS# = Student.*
WITH SIMILARITY > 0.8
```

The clause "where Faculty.SS# = Student.*" causes the Semantic Integration Process to find candidate corresponding attributes in the Student database, to be combined with Faculty.SS# as a global ID. The clause "WITH SIMILARITY >0.8" restricts candidate attributes to those that have a degree of similarity greater than 0.8. The query is processed in the following steps:

Step 1: Semantic Integration. Semint recommends the attribute correspondence as:

(Faculty.SS#, Student.Stud_ID, similarity = 0.98)

Step 2: Query Re-formulation. The "*" is replaced by the corresponding attribute (Student.Stud_ID) found in the previous step. However, if multiple corresponding attributes are recommended by Semint, one multidatabase query is generated for each corresponding attribute.

```
MULTISELECT *
FROM Faculty, Student
WHERE Faculty.SS# = Student.Stud_ID
```

Step 3: Multidatabase Query Processing. The Multidatabase query is translated into subqueries that are submitted to the component databases. The subqueries are executed at the component databases and the resulting tuples are returned to the originating site. Step 4: Data Integration with Ranked Result Sets. Because Semint only recommends one candidate GID, only one set of results is presented to the user. The result is shown in Figure 12.

Nothing is known about the Global ID. The query in the previous section would not give a correct result if there were no attribute in Student corresponding to Faculty.SS#. Other attributes, such as Name, may also work as a global ID. Therefore, we release the constraint that Faculty.SS# be part of the global ID. Instead, we let the system recommend candidate global IDs. The query is shown below:

```
MULTISELECT *
FROM Faculty, Student
WHERE Faculty.* = Student.*
WITH SIMILARITY > 0.8
```

The clause "WHERE Faculty.* = Student.*" specifies that the Semantic Integration Process should use any pair of corresponding attributes as candidate global IDs. Note that no similarity threshold clause is specified, so a default is used to restrict the candidates. The query is processed in the following steps:

Step 1: Semantic Integration. Semint recommends the following attribute correspondences:

(Faculty.SS#, Student.Stud_ID, similarity = 0.98)
(Faculty.Facu_Name, Student.Stud_Name, similarity = 0.91)
(Faculty.Salary, Student.Stipend, similarity = 0.85)

Step 2: Query Re-formulation. The "*"s are replaced by the attribute correspondences generated in the Step 1. Semint recommends three pairs of corresponding attributes; therefore, three multidatabase queries are generated as follows:

```
MULTISELECT *
FROM Faculty, Student
WHERE Faculty.SS# = Student.Stud_ID

MULTISELECT *
FROM Faculty, Student
WHERE Faculty.Facu_Name = Student.Stud_Name

MULTISELECT *
FROM Faculty, Student
WHERE Faculty.Salary = Student.Stipend
```

Step 3: Multidatabase Query Processing. These three multidatabase queries are translated into subqueries and submitted to the component databases. The component databases process these subqueries and return the results to the originating site.

Step 4: Data Integration. The Data Integration Process then merges the corresponding subquery results into integrated results for those originating queries using GID (Faculty.SS#, Student.Stud_ID), GID (Faculty.Facu_Name, Student.Stud_Name), and GID (Faculty.Salary, Student.Stipend). The results of these three queries are shown

(Facu_Name=Stud_Name)	Stud_ID	Stipend	SS#	Salary	Tel#
Larry	485-95-6784	$20,000	485-95-6784	$12,000	767-0900
Patricia	849-45-0500	$60,000	485-75-2374	$0	767-5134

Figure 13: Result using (Facu_Name,Stud_Name) as GID with Degree of Confidence = 0.91

(Salary=Stipend)	SS#	Facu_Name	Stud_ID	Stud_Name	Tel#

Figure 14: Result using (Salary,Stipend) as GID with Degree of Confidence = 0.85

in Figures 12-14 respectively.

Step 5: Data Integration with Ranked Result Sets. The integrated query results using three candidate GIDs, three tables, are then presented to the user as "ranked result sets". The degree of confidence of a result table is based on the similarity of corresponding attributes used as GID.

In this example, the Data Integration Process generates three sets of results that comprise the possible answers to the above query: Result of using (SS#,Stud_ID) as GID with Confidence = 0.98, result using (Facu_Name,Stud_Name) as GID with Confidence = 0.91, and result using (Salary,Stipend) as GID with Confidence = 0.85. Only at this point does the user need to perform "schema integration", by deciding which of the results is correct. If we know that there are some student instructors, we can see that the result using Salary and Stipend as the global ID is clearly incorrect, as there are no tuples in the Faculty/Student result table.

Choosing between (Facu_Name,Stud_Name) and (SS#,Stud_ID) is more difficult, as we have a similar result for both. However, using outside knowledge (such as student-instructors receive a much lower salary than Faculty, or after seeing Student ID we realize it is the same as social security number), we could surmise that Patricia is probably not a student-instructor. Therefore (SS#,Stud_ID) is a more likely global ID (the fact that this gives only tuples where BOTH the name and ID are the same supports this conjecture). Thus, the user is able to use additional information to resolve heterogeneity, information that was not available before the query was issued.

Correspondences other than join criteria

There are attribute correspondences other than Global IDs, such as aggregation, union, and intersection. These result in the additional problem of data integration: How do we combine different attributes into a single value (do we add them, average them, choose one as correct, or something else)? There are a number of approaches to this problem, described earlier. Each of these approaches leads to differences in how to specify queries (and thus in the query language), therefore we do not specify yet another query language for these cases. The dynamic schema integration process presented here can incorporate any of the dynamic data integration methods referenced.

We will give an example, however. If we assume an explicit aggregation technique, we could specify a query selecting total income of Faculty and Students as follows:

```
MULTISELECT Faculty.Salary+Student.*
FROM Faculty, Student
WHERE  Faculty.SS# = Student.*
WITH SIMILARITY > 0.8
```

This would generate tables showing each possible match for Faculty salary, for each possible match for Faculty SS#. Although this gives a potentially large number of potential tables (the product of the number of matches for each unspecified attribute), the required effort of choosing which of the tables is correct (given that most will be empty or have obviously nonsensical results) is substantially less than manually determining the correct attribute correspondence in advance.

Conclusions

In this paper we argue that attribute correspondence identification is the bottleneck of multidatabase dynamic integration and query processing. We have presented the theoretical background and design of Semint. We show how Semint can automatically extract metadata from databases in different DBMSs and determine attribute correspondences among databases. Semint uses only information that can be automatically extracted from a database. One novel feature is that how this metadata characterizes attribute semantics is learned, not pre-programmed. This allows Semint to automatically develop different "solutions" for different databases. The means of performing semantic integration is customized to the domain of the databases being integrated. This is done without human (or other outside) intervention; the "domain knowledge" is learned directly from the database. This enables us to find attribute correspondences between databases without providing any advance domain knowledge; rather than substantial *a priori* effort to codify domain knowledge, users need only use their domain knowledge to verify the recommendations of Semint.

We then demonstrated how Semint can be part of semiautomated integration in assisting DBAs in constructing attribute correspondences. We have used an example to dem-

onstrate the procedure. However, with Semint, we can go beyond this to perform dynamic integration: Determining the mappings that combine information from different databases after the query is issued. We have presented a method that uses ranked result sets to present query results to the user based on likely attribute correspondences between the databases. The user is still responsible for final determination of the correct mapping (a task requiring domain knowledge); but additional information, the query results and the ranking of these results, is provided to ease this task. Often the correct choice of attribute correspondence and means of mapping data will be obvious once the results are shown.

Note that other highly automated attribute correspondence techniques, such as [Housman, 1994], can be used in this same dynamic integration framework (or better still, the results from such tools can be combined) with those from Semint). We are currently working on this. A variety of data integration methods can also be incorporated in this framework. We are investigating using role sets (Scheuermann and Chong, 1994) to develop a complete dynamic integration system.

Acknowledgments

We would like to thank Peter Scheuermann for comments and suggestions in developing the ideas in this work. We also want to thank the referees for their suggestions on this manuscript.

Endnotes

[1] This material is based upon work supported by the National Science Foundation under Grant No. CCR-9210704.

[2] The views and opinions in this paper are those of the author and do not reflect MITRE's work position.

[3] However, users can still submit a multidatabase query to join Faculty and Student.

[4] This does not require retraining the network, only using the existing network. The parsing, classification, and training steps can be done in advance on a per-database basis without knowing what databases might be integrated with in the future; the only run-time processing is using an already trained network. This is a sub-second process, and will not noticeably affect response time).

[5] With more than two attributes, candidate GIDs are those that are similar among all the databases.

[6] For the Faculty and Student candidate keys in the example of Figure 8, three queries are generated.

References

Batini et al., 1986 Batini, C., Lenzerini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323—364.

Bright et al., 1994 Bright, M. W., Hurson, A. R., and Pakzad, S. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, 19(2):212—253.

Goh et al., 1994 Goh, C. H., Madnick, S. E., and Siegel, M. D. (1994). Context interchange: Overcoming the challenges of large-scale interoperable database system. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pages 337—346. ACM.

Hayne and Ram, 1990 Hayne, S. and Ram, S. (1990). Multi-user view integration system (MUVIS): An expert system for view integration. In *Proceedings of the 6th International Conference on Data Engineering*, pages 402—409. IEEE.

Housman, 1994 Housman, E. M. (1994). Data element analysis for wing level tbm c2 systems. Technical Report MTR 94B0000002, The MITRE Corporation, Bedford, MA.

Hwang et al., 1984 Hwang, H., Dayal, U., and Gouda, M. (1984). Using semiouterjoins to process queries in multidatabase systems. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 153—162. ACM.

Li and Clifton, 1994 Li, W.-S. and Clifton, C. (1994). Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 1—12, Santiago, Chile. VLDB.

Li and Clifton, 1995 Li, W.-S. and Clifton, C. (1995). Semint: A system prototype for semantic integration in heterogeneous databases. In *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, California.

Litwin et al., 1989 Litwin, W., Abdellatif, A., Zeroual, A., Nicolas, B., and Vigier, P. (1989). MSQL: A multidatabase language. *Information Sciences*, 49:59—101.

McLeod and Heimbigner, 1980 McLeod, D. and Heimbigner, D. (1980). A federated architecture for database systems. In *Proceedings of the National Computer Conference*, pages 283—289, Anaheim, CA. AFIPS.

Parent and Spaccapietra, 1995 Parent, C. and Spaccapietra, S. (1995). Database integration: An overview of issues and approaches. Submitted to *Communications of the ACM*.

Premerlani and Blaha, 1994 Premerlani, W. J. and Blaha, M. R. (1994). An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5):42—49.

Rosenthal and Reiner, 1984 Rosenthal, A. and Reiner, D. (1984). Extending the algebraic framework of query processing to handle outerjoins. In *Proceedings of 10th International Conference on Very Large Data Bases*, pages 334—343.

Scheuermann and Chong, 1994 Scheuermann, P. and Chong, E. I. (1994). Role-based query processing in multidatabase systems. In *Proceedings of the International Conference on Extending Database Technology*, pages 95—108.

Sheth and Kashyap, 1992 Sheth, A. and Kashyap, V. (1992). So far (schematically) yet so near (semantically). In *Proceedings of the IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems*, Victoria, Australia.

Sheth et al., 1988 Sheth, A., Larson, J., Cornelio, A., and Navathe, S. B. (1988). A tool for integrating conceptual schemas and user views. In *Proceedings of the 4th International Conference on Data Engineering*, Los Angeles, CA. IEEE.

Ventrone and Heiler, 1994 Ventrone, V. and Heiler, S. (1994). Some advice for dealing with semantic heterogeneity in federated database systems. In *Proceedings of the Database Colloquium*, San Diego. AFCEA (Armed Forces Communications and Electronics Assc.).

Wiederhold, 1992 Wiederhold, G. (1992). The roles of artificial intelligence in information systems. *Journal of Intelligent Information Systems*, 1(1):35—55.

Wen-Syan Li has recently joined the Center for Information Management, Integration, and Connectivity (CIMIC) at Rutgers University in August 1995. During the time the work described in this paper was being performed, he was obtaining his Ph.D. in the Department of Electrical Engineering and Computer Science at Northwestern University. Wen-Syan also holds an M.B.A. in Finance. His current research interests include heterogeneous databases, medical information systems, digital libraries. and AI applications. He can be reached at CIMIC, Rutgers University, 180 University Ave, Newark, NJ 07102, or by email at wsli@andromeda.rutgers.edu.

Chris Clifton has recently joined The MITRE Corporation as a Lead Database Systems Scientist in the National Intelligence Division. During the time the work described in this paper was being performed, he was an Assistant Professor in the Department of Electrical Engineering and Computer Science at Northwestern University. His research interests include heterogeneous database, data mining, and database support for text. He has published several conference papers and journal articles in these areas. Chris has a Ph.D. from Princeton University, and Bachelor's and Master's degrees from the Massachusetts Institute of Technology. He can be reached at The MITRE Corporation M/S K308, 202 Burlington Road, Bedford, MA 01730-1420, or by email at clifton@mitre.org.