Chapter 16

Integrating Data Sources Using A Standardized Global Dictionary

Ramon Lawrence and Ken Barker*

Department of Computer Science, University of Manitoba, Canada

* Department of Computer Science, University of Calgary, Canada

Key words: database, standard, dictionary, integration, schema, relational, wrapper, mediator, web

Abstract: With the constantly increasing reliance on database systems to store, process, and display data come the additional problems of using these systems properly. Most organizations have several data systems that must work together. As data warehouses, data marts, and other OLAP systems are added to the mix, the complexity of ensuring interoperability between these systems increases dramatically. Interoperability of database systems can be achieved by capturing the semantics of each system and providing a standardized framework for querying and exchanging semantic specifications.

Our work focuses on capturing the semantics of data stored in databases with the goal of integrating data sources within a company, across a network, and even on the World-Wide Web. Our approach to capturing data semantics revolves around the definition of a global dictionary that provides standardized terms for referencing and categorizing data. These standardized terms are then stored in record-based semantic specifications that store metadata and semantic descriptions of the data. Using these semantic specifications, it is possible to integrate diverse data sources even though they were not originally designed to work together.

A prototype of this integration system called the Relational Integration Model (RIM) has been built. This paper describes the architecture and benefits of the system, and its possible applications. The RIM application is currently being tested on production database systems and integration problems.

1. INTRODUCTION

Interoperability of database systems is becoming increasingly important as organizations increase their number of operational systems and add new decision-support systems. The construction, operation, and maintenance of these systems are complicated and time-consuming and grow quickly as the number of systems increases. Thus, a system that simplifies the construction and integration of data sources is of great importance.

Our work attempts to standardize the description of information. Behind most web sites and applications is a database storing the actual data. Our goal is to capture the semantics of the data stored in each database so that they may operate together. Capturing data semantics and integrating data sources is applicable to companies and organizations with multiple databases that must interoperate. More importantly, by presenting a framework for capturing data semantics, it is more likely that databases that were never intended to work together can be made to interoperate. This is especially important on the World-Wide Web, where users want to access data from multiple, apparently unrelated sources.

This work outlines a model for capturing data semantics to simplify the schema integration problem. The major contribution of the work is a systemized method for capturing data semantics using a standardized global dictionary and a model that uses this information to simplify schema integration in relational databases. This chapter describes the structure of the global dictionary and the benefits of the Relational Integration Model (RIM). Section 2 discusses the integration problem, and how capturing data semantics is fundamental to its solution. Previous work in the area is detailed in Section 3. Section 4 overviews our integration architecture, which utilizes a global dictionary (Section 5) for identifying similar concepts, and a record-based integration language, the Relational Integration Model (Section 6), used to exchange metadata on systems. Details on integration related problems are given in Section 7, and applications of our integration technique are presented in Section 8. The chapter closes with future work and conclusions.

2. DATA SEMANTICS AND THE INTEGRATION PROBLEM

Integrating data sources involves combining the concepts and knowledge in the individual data sources into an integrated view of the data. The integrated view is a uniform view of all the knowledge in the data sources so that the user is isolated from the individual system details. By isolating the user from the data sources and the complexity of combining their knowledge, systems become "interoperable", at least from the user's perspective, as they can access the data in all data sources without worrying about how to accomplish this task.

Constructing an integrated view of many data sources is difficult because they will store different types of data, in varying formats, with different meanings, and will be referenced using different names. Subsequently, the construction of the integrated view must, at some level, handle the different mechanisms for storing data (structural conflicts), for referencing data (naming conflicts), and for attributing meaning to the data (semantic conflicts).

16. Integrating Data Sources Using A Standardized Global Dictionary

Although considerable effort has been placed on integrating databases, the problem remains largely unsolved due to its complexity. Data in individual data sources must be integrated at both the schema level (the description of the data) and the data level (individual data instances). This chapter will focus on schema-level integration. Schema integration is difficult because at some level both the operational and data semantics of a database need to be known for integration to be successful.

The schema integration problem is the problem associated with combining diverse schemas of different databases into a coherent integrated view by reconciling any structural or semantic conflicts between the component databases. Automating the extraction and integration of this data is difficult because the semantics of the data are not fully captured by its organization and syntactic schema.

Integrating data sources using schema integration is involved in constructing both a multidatabase system and a data warehouse. Both of these architectures are finding applications in industry because they allow users transparent access to data across multiple sites and provide a uniform, encompassing view of the data in an organization. On an even wider-scale, a standardized mechanism for performing schema integration would allow a user's browser to automatically combine data from multiple web sites and present it appropriately to the user. Thus, a mechanism for performing schema integration would be of great theoretical and practical importance.

The literature has proposed various methods for integrating data sources. However, the fundamental problem in these systems is the inability to capture data semantics. Automated integration procedures cannot be applied without a systematic way of capturing the meaning of the stored data. In this work, we propose a method for capturing data semantics which bridges the theoretical work and the pragmatic approach used in industry. A standardized global dictionary is defines words to reference identical concepts across systems. We then demonstrate a systematic method of storing data semantics using these dictionary words for integrating relational schemas.

3. PREVIOUS WORK

The integration problem involves combining data from two or more data sources and is often required between databases with widely differing views on the data and its organization. Thus, integration is hard because conflicts at both the structural and semantic level must be addressed. Further complicating the problem is that most systems do not explicitly capture semantic information. This forces designers performing the integration to impose assumptions on the data and manually integrate various data sources based on those assumptions. To perform integration, some specification of data semantics is required to identify related data. Since names and structure in a schema do not always provide a good indication of data meaning, it often falls on the designer to determine when data sources store related or equivalent data. The integration problem is related to the database view integration problem¹. Batini² surveys early manual integration algorithms.

Previous work in this area has focused on capturing metadata about the data sources to aid integration. This metadata can be in the form of rules such as the work done by Sheth³, or using some form of global dictionary such as work done by Castano⁴. We believe that the best approach involves defining a global dictionary rather than using rules to relate systems because rules are more subject to schema changes than a global dictionary implementation and grow exponentially as the number of systems increase.

Other research efforts include the definition of wrapper and mediator systems such as Information Manifold⁵ and TSIMMIS⁶. These systems provide interoperability of structured and unstructured data sources by "wrapping" data sources using translation software. Once a data source has been integrated into the overall system, distributed querying is possible. However, the construction of the global view is mostly a manual process. Our approach is complimentary to these systems by studying how standardized dictionaries can be used to simplify and automate the construction of the integrated or global view.

Work on capturing metadata information in industry has resulted in the formation of a metadata consortium involving many companies in the database and software communities. The goal of the consortium is to standardize ways of capturing metadata so that it may be exchanged between systems. The consortium has defined the Metadata Interchange Specification (MDIS) version 1.1⁷ as an emerging standard for specifying and exchanging metadata. The structured format is very good at specifying the data structure, names, and other schema information in record-form. However, the only method for capturing data semantics, besides the schema names used, is by using text description fields or storing semantic names in long name fields. The problem with this method is that systems cannot automatically process these optional description fields to determine equivalent data using the metadata information. Basically, this system lacks a global dictionary to relate terms.

Another emerging metadata standard is Extensible Markup Language $(XML)^8$. The power of XML as a description language is its ability to associate markup terms with data elements. These markup terms serve as metadata allowing formalized description of the content and structure of the accompanying data. Unfortunately, the use of XML for integration is limited because XML does not define a standardized dictionary of terms. Thus, our approach extends industrial methodologies by systematically defining and applying a dictionary of terms that can be used across domains, industries, and organizations.

Global dictionaries have been used before to perform integration. Typically, the dictionary chosen consists of most (or all) of the English language terms. For example, the Carnot Project⁹ used the Cyc Knowledge base as a global dictionary. The problem with such large dictionaries is that they lead to ambiguity that further complicates integration.

4. THE INTEGRATION ARCHITECTURE

Our integration architecture consists of two separate and distinct phases: the *capture process* and the *integration process*.

In the capture process (see Figure 1), the semantics and the metadata of a given data source are represented in record form using terms extracted from a global dictionary. This capture process is performed independently of the capture processes that may be occurring on other data sources. The only "binding" between individual capture processes at different data sources is the use of the global dictionary to provide standardized terms for referencing data.

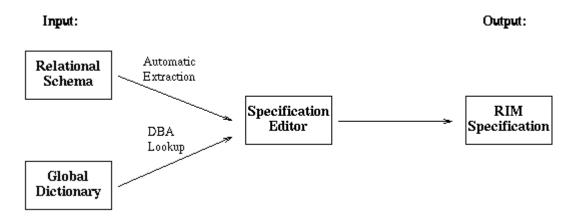


Figure 1. The Capture Process on a Relational Database

The inputs of the capture process are a relational database and the global dictionary. Using a customized tool called a specification editor, the database administrator (DBA) attempts to capture the data semantics in a record-based form. The records storing the data description are called RIM specifications (explained in Section 6).

From the relational database schema, the specification editor extracts typical metadata information such as table names, field sizes and types, keys, indices, and constraints. All this information is readily available in the database schema and does not require any input by the DBA. This schema information is then stored in the appropriate fields in the RIM specification.

The second step in the capture process is less automatic. While the specification editor is parsing the database schema, it attempts to match attribute and table names to entries in the global dictionary. Since field names in schemas are not always descriptive, this matching process works by comparing the field names on a best effort basis to existing global dictionary terms. In addition, as system field names are matched to their global counterparts, this information is stored in the global dictionary for future use. Basically, the system remembers mappings between system names and global concepts, so if a system name is encountered again, the mapping is performed automatically.

The system's initial attempt at mapping system names to global concepts will almost always miss the semantic intent of some fields. It then falls on the DBA to determine the correct global dictionary terms for all concepts in the database. Depending on the size of the database, this may be a time consuming task, but it only needs to be performed once.

After all system names are mapped to terms in the global dictionary, these global terms are stored in the RIM specifications along with the schema information. The output of the capture process is a complete RIM specification that contains the necessary information to describe the data stored in the data source and to integrate it with other data sources. These RIM specifications are similar to export schemas in federated systems¹⁰ with one difference; they also contain sufficient metadata to automate the integration process.

The integration phase of the architecture (see Figure 2) actually performs the integration of various data sources. The integration process begins when a client accesses two or more data sources. To access the data sources, a client connects and is authenticated by each data source required. The data source provides the client with its RIM specification describing its data. These RIM specifications are then combined at the client site by an integration algorithm that detects and resolve structural and semantic conflicts between the data sources. The output of the integration algorithm is an integrated view on which the client poses queries.

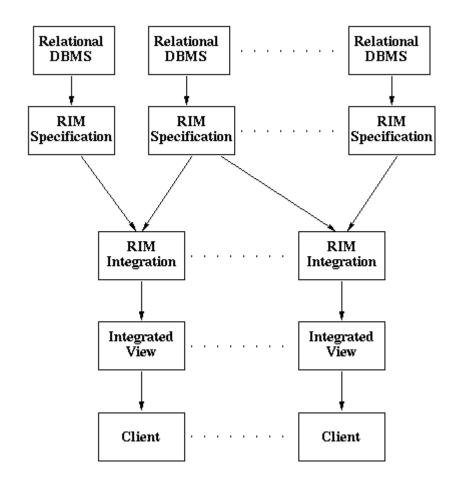


Figure 2. The Integration Process for Relational Databases

The integration algorithm is responsible for combining the RIM specifications for each data source and for partitioning a query posed on the integrated view into subtransactions on the individual data sources (de-integration). A brief description of the integration algorithm is provided in Section 6.

The two-phase architecture has several desirable properties:

- 1. Performs dynamic integration schemas are combined one at a time into the integrated view
- 2. Captures both operational and structural metadata and uses it to resolve conflicts

- 3. Performs automatic conflict resolution
- 4. Metadata is captured only at the local level and semi-automatically which removes the designer from most integration-related problems
- 5. Local schemas are merged into an integrated view using a global dictionary to resolve naming conflicts

The key benefit to the two-phase process is that the capture process is isolated from the integration process. This allows multiple capture processes to be performed concurrently and without knowledge of each other. Thus, the capture process at one data source does not affect the capture process at any other data source. This allows the capture process to be performed only once, regardless of how many data sources are integrated. This is a significant advantage as it allows application vendors and database designers to capture the semantics of their systems at design-time, and the clients of their products are able to integrate them with other systems with minimum effort.

The underlying framework which ties these capture processes together is the RIM specification. The RIM specification contains the schema information and semantic terms from the global dictionary. Using these terms, the integration algorithm is able to identify similar concepts even though their structural representations may be very different. As the global dictionary and its structure are central to the integration architecture, it is discussed in detail in the following sections.

5. THE GLOBAL DICTIONARY

To provide a framework for exchanging knowledge, there must be a common language in which to describe the knowledge. During ordinary conversation, people use words and their definitions to exchange knowledge. Knowledge transfer in conversation arises from the definitions of the words used and the structure in which they are presented. Since a computer has no built-in mechanism for associating semantics to words and symbols, an on-line dictionary is required to allow the computer to determine semantically equivalent expressions.

The problem in defining a global dictionary is the complexity of determining semantically equivalent words and phrases. The English language is very large with many equivalent words for specifying equivalent concepts. Thus, using an on-line English dictionary for the computer to consult is not practical. Not only is the size of the database a problem, but it is complicated for the computer to determine when two words represent semantically equivalent data. Bright used an English language dictionary in defining the Summary Schemas Model (SSM)¹¹ to query multidatabases using imprecise words, but it is difficult to base an integration methodology on such a model.

The other alternative is to construct a global dictionary using words as they appear in database schema. This has the advantage that the global dictionary is only storing the required terms. However, it still may be difficult to integrate global dictionaries across systems that are derived in this manner depending on the exact words chosen to represent the data. Castano⁴ studied creating a global dictionary during integration.

Our approach is a hybrid of the two methodologies. The basis of the shared dictionary is a standardized concept hierarchy containing hypernym links relating concepts. This dictionary contains the most common concepts stored in databases. Each node in the hierarchy consists of a default term and definition. Synonyms for the default term in the node are also provided. The vast

majority of the concepts in a database including dates, addresses, names, id/key fields, and description fields are captured using this simplified hierarchy.

The second component of the global dictionary is component relationships. Component relationships relate terms using a 'Part of or 'HAS A' relationship. For example, an address has (or may have) city, state, postal code, and country components. Similarly, a person's name may have a first, last, and full name components. These component relationships are intended to standardize how common concepts with subcomponents are represented.

Ideally, a global dictionary consisting of a concept hierarchy and a set of component relationships would be enough to represent all data in databases. However, this is not possible as new types of data and very specialized data would not appear in the dictionary. Although a standards organization could continually evolve the global dictionary, this would not occur rapidly enough. Thus, we allow an organization to add nodes to the global dictionary to both the concept hierarchy and component relationships to capture and standardize names used in their organization that are not in the standardized global dictionary. These additional links are stored in a record format and are transmitted along with the metadata information during integration.

5.1 Using the Global Dictionary

The global dictionary serves as a mechanism for agreeing on the vocabulary describing the data. Without a standardized dictionary, it would be next to impossible to determine the meaning of data. The dictionary provides the computer with a framework for comparing individual words across database systems to determine if they are semantically equivalent. Obviously, when hypernym links are involved, there must be some mechanism for defining partial equivalence; terms that are related but not exactly similar. For this paper, we will use an approach proposed by Castano⁴. The semantic similarity between two words in the dictionary is based on the number of hypernym links between them. Each hypernym link has a value of 0.8, so traversing two hypernym links would result in a semantic similarity of 0.64.

Using a single word to describe the semantics of a database field or relation is insufficient. It is highly unlikely that a given dictionary word is able to capture all the semantics of a database element. We are proposing capturing data semantics using a semantic name with the following structure:

Semantic Name = $[CT_1; CT_2; ...; CT_N]$ Concept name where CT_i is a context term

A context term is used to define the general context to which the semantic term applies, and typically corresponds to an entity or a relationship in the database. It is possible to have a hierarchy of context terms. For example, the city field of an address of a person has two contexts: the person and the person's address (i.e. [Person;Address] City). The concept name is a dictionary term that represents the semantics of the attribute describing the context. A semantic name does not have to include a concept name. Typically, a semantic name will contain a concept name only if it is describing an attribute of a table. A semantic name will not have a concept name if it is defining a context. Context terms and concept names are either extracted directly from the global dictionary or added to the global dictionary as needed.

9

The semantic names defined using this procedure can then be included in a metadata specification such as MDIS version 1.1 (in the description field), a RIM specification, as a XML tag, or another model capturing semantic metadata using records. The system will be able to use this information more readily to integrate metadata automatically than a plain-text description field.

5.2 Comparing semantic names

The first step in integrating RIM specifications is identifying equivalent or semantically similar names. The semantic distance between the two terms in the dictionary can measure the semantic similarity of two terms. The definition of semantic distance can vary by application. In some cases, two semantic names may be only deemed equivalent if they are exactly identical. For the purpose of this paper, we will define a simple formula for semantic distance based on the number of hypernym links separating the terms in the dictionary. However, other definitions of semantic distance are equally valid and may be more effective in certain applications.

Given a global dictionary D, a set of added global terms A, and two metadata specifications M_1 and M_2 , an automated integration system must take the two specifications and use the dictionary to construct an integrated schema. Ideally, this process would be performed with minimal user input, although some user tuning may have to be performed at the end of the process. The first step is identifying semantically similar terms in the two specifications.

Let $F_{SD}(T_1, T_2)$ be a function that computes the semantic distance between two terms T_1 and T_2 in the dictionary. $F_{SD}(T_1, T_2) = 0.8^N$, where N = Number of hypernym links separating T_1 and T_2 . For any semantic names S_1 from M_1 and S_2 from M_2 , the system determines their semantic equivalence by first matching context terms then comparing concept names (if present).

The first step in comparing two semantic names S_1 and S_2 is to separate each semantic name S_i into a set of N context terms, $CT_{i(1..N)}$, and a concept name CN_i . Then, given the two sets of context terms $CT_{I(1..N)}$ and $CT_{2(1..M)}$, the individual global dictionary terms are compared in the order they appear.

Assume that S_1 is a semantic name already in the Integrated View (IV), and S_2 is the semantic name to be integrated into the IV. The distance between the two contexts is given by:

$$CT_{SD} = \prod_{i=1..min(N,M)} F_{SD}(CT_{1(i)}, CT_{2(i)})$$

That is, the semantic distance between the two contexts is the product of the semantic distances between the individual context terms taken pairwise in the order that they appear. If the contexts are sufficiently similar, then the concept names are compared. The semantic distance between two concept names is given by: $CN_{SD} = F_{SD}(CN_1, CN_2)$.

Thus, the unmodified semantic distance between S_1 and S_2 is $U_{SD} = CT_{SD} * CN_{SD}$. The unmodified semantic distance is not the final semantic distance calculated. Since the integration algorithm will be comparing the new semantic term S_2 to all semantic terms in the integrated view, modification constants are used to indicate more desirable matches in addition to semantic similarity in the global dictionary.

For example, assume $S_2 = [A;B] C$. If $S_1 = [A;B;C]$, $U_{SD} = 1$ as the first two contexts (A,B) are identical and S_1 and S_2 do not both have concept names to compare. However, modification constants are applied to "penalize" S_1 for having an extra context (C) and no concept name. These constants insure that if $S_3 = [A;B] C$, then the semantic similarity of S_2 and S_3 is 1, while the similarity of S_1 and S_3 would be slightly less than 1.

6. THE RELATIONAL INTEGRATION MODEL (RIM)

Using semantic names and the global dictionary, the system can more readily determine identical concepts in a metadata specification than by using only plain-text description fields or logical names. These semantic names can be inserted into a metadata specification model such as MDIS version 1.1⁷, but we have also defined our own model for capturing and specifying metadata.

The Relational Integration Model (RIM) is designed to capture metadata and semantic names to allow automatic integration of relational database systems. The core idea behind the model is that semantic names are used to determine equivalent concepts across metadata specifications, and the model has been designed to represent schema in a way that makes it easy to convert the structure of the data as needed during integration.

The system is based on the ER model and models data as an entity, relationship, or attribute. A semantic name is associated with all entities and attributes (and optionally relationships), so that they may be identified during integration. Once identical (or semantically similar) concepts are found, it is easy to transform the concept's structural representation. For example, if an identical concept is represented as an entity in one schema and a relationship in another, the concept can be represented as either an entity or a relationship in the integrated schema, and the other representation can be mapped or converted into the chosen one.

RIM is similar to MDIS as it captures semantic information in record-form. However, we are planning on expanding RIM to capture behavioral characteristics in the future, hence the departure from MDIS. Using RIM and capturing semantic names, we can identify related global concepts across systems and transform their structures into an integrated schema.

6.1 The Integration Algorithm

To produce an integrated view of data sources, RIM specifications are first constructed containing metadata on each data source and semantic names describing their data. These RIM specifications then must be combined using the global dictionary and an integration algorithm into an integrated view. The integration algorithm combining RIM specifications consists of two distinct phases:

- 1. **Matching phase:** combines the semantic terms in the specification using the global dictionary into an integrated view
- 2. **Metadata phase:** uses additional metadata from the individual data sources (including relationships, constraints, and comments) to refine the integrated view

The matching phase proceeds by taking a RIM specification M_1 and combining it with the integrated view. The algorithm can be abstracted as follows:

```
Procedure RIM integration (M as RIM spec., IV as Integrated View)
   For each RIM header (context) {\tt H}_{\rm i} in M
      sn = semantic name of H_i
      match_sname(sn,H<sub>i</sub>,IV)
   Next i
   For each RIM schema (concept) {\tt T}_{\rm i} in M
      sn = semantic name of T_i
      match_sname(sn,Ti,IV)
   Next i
End Procedure
Procedure match_sname(sn as semantic_name, V as Context/Concept, IV as Int_View)
   max = 0
   For all root-level contexts \ensuremath{C_{\mathrm{i}}} in the IV
      Recursive_match(sn,C<sub>i</sub>,best_match,sem_dist)
      If sem_dist > max Then Keep Best Match (max)
   Next
   // Add sn to IV: If there is a best_match use it, otherwise add new name
   Integrate_semantic_name(sn, best_match, IV)
End Procedure
Procedure Recursive match(sn as Semantic name, CT as Context,
    best_match as Variant, max as double)
   max = 0
   sd = Compute SD(sn, CT)
   cutoff_value = 0.5
   If sd > cutoff_value Then
      If sn has more context terms than CT Then
          For each subcontext SC<sub>1</sub> of CT
             Recursive_match(sn,SC<sub>i</sub>,best_match,sem_dist,status)
             If sem_dist > max Then Keep best match (max)
          Next
      End If
      For each concept CN_i in CT
          sd = Compute SD(sn, CN_i)
          If sd > max Then Keep best match (max)
      Next
   End If
End Procedure
```

It is important to note that each semantic name in the RIM specification is separately integrated into the integrated view. Thus, once a term from the specification is added to the view, the next term from the specification will also be integrated with the newly added term. This is desirable because the integration algorithm is consistent regardless of the number of RIM specifications combined into the integrated view, even if there are none. Another desirable feature is that integration occurs "within" a specification itself, which combines contexts and concepts into a better form than they originally appear. Thus, RIM headers (contexts) are added first, in order to provide a framework for the RIM schemas (concepts) to be integrated into.

The majority of the integration work is performed by the semantic name matching algorithm. This algorithm takes a semantic name to be integrated, and finds and combines the term appropriately into the integrated view. The matching algorithm recursively matches a given semantic name S_i to all contexts and concepts in the integrated view.

The *recursive_match* function takes the semantic name of S_i (**sn** in code), and the root-level context C_i , and recursively matches S_i with C_i and all of its subcontexts. The *recursive_match* function then returns the best matching context or concept, and its semantic distance from S_i . If a match is found, the type of match is important because it indicates if schema changes may be necessary. The type of match depends on if the integrated view term and S_i are contexts or concepts. For example, a context-concept type match occurs if the integrated view term is a context and S_i is a concept. Similarly, 3 other types of matches are possible: context-context, concept-concept, and concept-context.

At this point, the semantic name S_i has been compared to the integrated view, and its proper integration point has been determined. The *integrate_semantic_name* procedure takes this information and updates the view. The exact result will depend on if a match was made and its type. The procedure may add contexts, subcontexts, and concepts to the integrated view, especially when S_i did not perfectly match anything in the view. Regardless, this procedure records S_i in the integrated view and the database source information for use in querying. Replication of this procedure for all terms S_i in the RIM specification completes the integration.

Once all semantic terms of the RIM specification are initially integrated into the view, a second integration phase is performed. The second step uses metadata in the individual data sources to refine the integration. Semantic names are not used in this procedure. The metadata phase of the integration algorithm performs the following actions:

- 1. Uses relationship information such as cardinalities in the specification to promote/demote hierarchical contexts created in the initial phase.
- 2. Uses field sizes and types to validate initial matchings.
- 3. Creates views to calculate totals for data sources that are integrated with data sources containing totals on semantically similar data.

6.2 Integration Example using RIM

To illustrate the application of RIM specifications and a global dictionary, this section presents a very simple example of its use. For this example, a reduced RIM specification format will be used which consists only of the system names and their associated semantic names for each table and field in the databases. Normally, field sizes, types, indices, foreign keys, and other syntactic metadata is also present in a RIM specification. Consider XYZ shipping company that has a relational database storing damage claim information, and ABC hauling company that uses a similar database. XYZ shipping company just bought out ABC hauling company and would like to use both databases together instead of building a new one. The structure of the XYZ database is: *Claims_tb(claim_id, claimant, net_amount, paid_amount)*. The structure of the ABC database is: $T_claims(id, customer, claim_amount)$ and $T_payments(cid, pid, amount)$. Notice that the ABC database may store multiple payments per claim, whereas the XYZ database only stores one payment amount.

A capture process is performed on each database to produce a RIM specification describing their data. The RIM specifications for the XYZ database and the ABC database are given in Tables 1 and 2 respectively.

Туре	System Name	Semantic Name
Table	Claims_tb	[Claim]
Field	Claim_id	[Claim] Id
Field	Claimant	[Claim] Claimant
Field	Net_amount	[Claim] Net amount
Field	Paid_amount	[Claim;Payment] Amount

Table 1. RIM Specification for XYZ Database

Туре	System Name	Semantic Name
Table	T_claims	[Claim]
Field	Id	[Claim] Id
Field	Customer	[Claim] Customer
Field	Claim_amount	[Claim] Amount
Table	T_payments	[Claim;Payment]
Field	Cid	[Claim] Id
Field	Pid	[Claim;Payment] Id
Field	Amount	[Claim;Payment] Amount

Each table has a semantic name associated with the entity (record instance) it contains. (i.e. "Claim" and "Payment") A field of a table modifies or describes the entity, which is the context of the attribute. Thus, the *claim_id* field in *Claims_tb* has a context of "Claim" because it describes the claim entity, and a concept (or attribute) name of "id". In the case of the field *pid* in table *T_payments*, the id attribute actually describes two contexts: a payment and a claim, as a payment itself describes a claim.

It is obvious from examining the two tables that the integration algorithm would detect equivalent concepts for many fields using strict equivalence on the semantic names. However, in some cases a measure of semantic distance is required. For example, the semantic names [Claim] Claimant and [Claim] Customer are not equivalent, but would have a very low semantic distance as the two terms would be very close together in the dictionary.

The integrated view produced is: *Claim(id, claimant, amount, Payment(id, amount))*. Note that in the case of the XYZ company database the payment id is NULL as there is only one payment. Using the global dictionary the integrated view is produced automatically using the RIM specifications. The integrated view is a hierarchy of concepts rather than a physical structure. Thus, a mapping is performed during querying from semantic names representing concepts to the system names of physical fields and tables in the underlying data sources.

A major benefit of this procedure is that no explicit mappings or rules are produced relating the two systems. This means that the schemas of the systems can be changed without requiring the integration to be totally re-done. Only the RIM specification of the affected database would need to be changed, and the integration procedure should be able to automatically reconstruct an updated view that reflects the changes. Finally, integrating new data sources is significantly easier as only one new RIM specification would have to be created. Suppose CDE company bought out ABC and XYZ companies. To integrate all three databases, a RIM specification is only constructed for the CDE database, which is then combined with the two other databases without any knowledge of their structure. Thus, this architecture scales well as the number of databases to be integrated increases.

7. SPECIAL CASES OF INTEGRATION

With properly selected semantic terms, most integrations produce expected results. However, there are some special cases to consider. This section explains how to resolve some of these issues.

The most serious integration problem is false matchings between semantic terms. Consider the previous example. One database stores a claimant and the other stores a customer. In the global dictionary, the term claimant is one link (subtype) below customer, so the semantic distance between the two terms is 0.8. The system then combines these terms together, which is a desirable result.

However, consider how phone numbers will be integrated. In the global dictionary, the term phone number has subtypes home phone number, fax number, and cell number among others. If phone number is already in the integrated view, and fax number is added, the two concepts will be combined as they have a semantic distance of 0.8. Unfortunately, unlike the customer-claimant combination, phone and fax numbers are separate concepts, which should not be combined even though they are semantically similar.

This false integration problem is called **hierarchical mis-integration**. The reason for these faulty integrations is that two semantically similar, hierarchical concepts are combined together based on their high semantic similarity even though they are distinct, real-world concepts and should not be combined.

One possible solution to this problem is to add weighted links between terms in the dictionary instead of using a uniform link value. Thus, the link between phone and fax number may only be 0.4, whereas the link between phone number and home phone number may be 0.8. The problem with this approach is the loss of uniformity in the global dictionary, and the difficulty in assigning link values to insure correct integrations.

Our approach involves the detection and promotion of related, hierarchical terms. The general idea is if term A currently exists in the integrated view and term B which is a subtype of A is added, then term A is promoted to a context ([A]), and the concept [A] A is added. Finally, the concept B is inserted as [A] B.

For example, consider the previous case of phone and fax numbers. Let the integrated view contain the semantic name [Customer] Phone Number (a customer's phone number), and we wish to add the fax number of the customer. The concept [Customer] Phone Number gets promoted to a

context, i.e. [Customer;Phone Number], then, a concept phone number is inserted [Customer;Phone Number] Phone Number. Finally, the new concept [Customer;Phone Number] Fax Number is added. The final view is: [Customer], [Customer;Phone Number], [Customer;Phone Number] Phone Number, [Customer;Phone Number] Fax Number.

Thus, the phone and fax number concepts are kept separate, and a higher-level notion (that of all phone numbers regardless of usage) is also defined. In most cases, this hierarchical promotion feature is desirable for integration. However, in some cases it may be inappropriate. In the claims database example, the merger of customer and claimant in the two databases would produce an integrated view of: [Claim],[Claim;Customer],[Claim;Customer;Claimant]. Ideally, the integrated view should be:[Claim], [Claim;Claimant]. That is, the user is generally not interested in the distinction between a customer and a claimant, as the distinction is more terminology than semantics. Thus, although the user is still allowed to query only on the customer and claimant "levels", for most queries the users wants to see only the claimant "level" with all customer and claimant information merged into it. This demotion or merger activity is performed in the metadata phase of integration and is based on the relationships between the concepts/contexts.

Another integration challenge is the handling of totals and aggregates. In the claims database example, it may be possible that the payment amount in the XYZ database is actually the total amount of all payments, and not the value of one payment. In this case, it makes more sense to name the field [Claims;Total;Payment] Amount, so that it is not integrated with the individual payment amounts in the ABC database. This implies the system should calculate the payment totals for the ABC database as required.

Finally, in some cases a default concept name should be assumed for a semantic name, even if it already has a concept name. For example, the semantic name for a city in a database may be given as [Address] City, however this should really be represented as [Address;City] Name. Depending on the dictionary term, a default concept name of "name" or "description" may be appropriate.

8. APPLICATIONS TO THE WWW

Integrating data sources automatically would have a major impact on how the World-Wide Web is used. The major limitation in the use of the Net, besides the limited bandwidth, is in the inability to find and integrate the extensive databases of knowledge that exist. When a user accesses the Web for information, they are often required to access many different web sites and systems, and manually pull together the information presented to them. The task of finding, filtering, and integrating data consumes the majority of the time, when all the user really requires is the information. For example, when the user wishes to purchase a product on-line and wants the best price, it is up to the user to visit the appropriate web sites and "comparison shop". It would be useful if the user's web browser could do the comparison-shopping for them.

In our architecture, these types of queries are now possible. To achieve this, each web site would specify their database using a RIM specification. The client's browser would contain the global dictionary and a list of web sites to access to gather information. When the user wishes to purchase an item, the browser downloads the RIM specifications from the on-line stores, integrates them using the global dictionary, and then allows the user to query all databases at once through the

"integrated view of web sites" that was constructed. Obviously, the integration itself is complex, but a system that achieves automatic integration of data sources would have a major impact on how the Web is used and delivered.

9. FUTURE WORK AND CONCLUSIONS

In this chapter, we have detailed how a standardized global dictionary, a formalized method for capturing data semantics, and a systematic approach for calculating semantic distance between phrases can be used to identify similar concepts across data sources. These semantic names can then be inserted into a metadata descriptive model such as MDIS or RIM, and can be automatically processed during integration. This reduces the amount of user input required to integrate diverse data sources and serves as a mechanism for capturing and displaying semantics of the data.

Future work includes refining the standardized dictionary and modifying the integration software. We have already completed construction of RIM software including a specification editor tool and integration module. We plan to test its performance on existing systems and integration problems in the near future.

REFERENCES:

- **1.** R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996, pages 481-492.
- **2.** C. Batini, M. Lenzerini and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4) December 1986, pages 323-364.
- **3.** A. P. Sheth and G. Karabatis. Multidatabase interdependencies in industry. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Washington, DC, USA, 1993, pages 481-492.
- 4. S. Castano and V. Antonellis. Semantic dictionary design for database interoperability. In Proceedings of the 13th International Conference on Data Engineering (ICDE'97), 1997, pages 43-54.
- 5. T. Kirk, A. Levy, Y. Sagiv and D. Srivastava. The Information Manifold. In AAAI Spring Symposium on Information Gathering, 1995.
- 6. C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman and M. Valiveti. Capability Based Mediation in TSIMMIS. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Seattle, WA, USA, 1998, pages 564-566.
- 7. The Metadata coalition. Metadata interchange specification. Technical Report version 1.1, August 1997.
- 8. W3C. Extensible Markup Langauge (XML) specification. Technical Report version 1.0, February 1998.
- **9.** C. Collet, M. Huhns, and W-M. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*. 24(12), December 1991, pages 55-62.
- 10. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogenous and autonomous databases. *ACM Computing Surveys*, 22(3) September 1990, pages 183-236.
- M. Bright, A.Hurson, and S. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. In ACM Transactions on Database Systems, 19(2) June 1994, pages 212-253.