Automatic Integration of Relational Database Schemas[1]

University of Calgary 2000-662-14
University of Manitoba TR-00-15

Ramon Lawrence
Advanced Database Systems Laboratory
Department of Computer Science
University of Manitoba
umlawren@cs.umanitoba.ca

Ken Barker
Advanced Database Systems and Applications Laboratory
Department of Computer Science
University of Calgary
barker@cpsc.ucalgary.ca

Abstract

This paper 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 standardized dictionary which provides terms for referencing and categorizing data. These standardized terms are then stored in semantic specifications called X-Specs which store metadata and semantic descriptions of the data. Using these semantic specifications, it becomes possible to integrate diverse data sources even though they were not originally designed to work together.

The centralized version of the architecture is presented which allows for the independent integration of data source information (represented using X-Specs) into a unified view of the data. The architecture preserves full autonomy of the underlying databases which are transparently accessed by the user from a central portal. Distributing the architecture would by-pass the central portal and allow integration of web data sources to be performed by a user's browser. Such a system which achieves automatic integration of data sources would have a major impact on how the Web is used and delivered.

Unlike wrapper or mediator systems which achieve data source integration by manually defining an integrated view, our architecture automatically constructs an integrated view from information independently provided by the data sources. Thus, the contribution is an algorithm for schema integration not just a methodology for accessing data sources whose knowledge has been precombined into mediated views. The integrated view is a hierarchy of concepts that is queried by semantic name. Thus, the system provides both logical and physical access transparency by mapping user queries on high-level concepts to physical schema elements in the underlying data sources.

1 Introduction

Although numerous architectures are proposed to provide database interoperability, automatic schema integration has not been previously possible. We propose that automatic schema integration is possible by using a standard dictionary of terms to describe schema element semantics. Effectively, the use of the dictionary resolves naming problems which allows our algorithm to automatically resolve the more complex structural and semantic conflicts. The major contribution of the work is a systemized method for capturing data semantics using a standardized dictionary and a model which uses this information to perform schema integration in relational databases. XML is used as a specification language for exchanging metadata on database schemas.

This paper describes the integration architecture and compares it with existing work. Section 2 discusses the integration problem and related work. Section 3 overviews the

[1]This research is partially sponsored by a Natural Science and Engineering Research Grant (RGP-0105566)

components of our integration architecture which utilizes a standard dictionary for identifying similar concepts, a specification language (X-specs) used to exchange metadata on systems, and an integration algorithm for combining the metadata specifications together into a unified schema. The integration architecture is discussed in Section 4. The architecture achieves automatic integration of diverse data sources while preserving their full autonomy. By exploiting the XML standard to exchange schema metadata and employing a standardized dictionary of terms, integration of data sources is provided by a central portal allowing the user to have transparent access to all data sources. Section 5 briefly overviews issues in querying the integrated view, mapping from the global view to local views, and supporting multiple languages. Finally, applications of the architecture are discussed, and the architecture is compared to mediator-based systems in Section 6. The paper 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 which isolates users from the individual system details. By isolating the user from the data sources and the complexity of combining their knowledge, systems become interoperable 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 global 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).

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 paper will focus on schema-level integration. The schema integration problem is the problem associated with combining the diverse schemas of the 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 syntatic schema.

Integrating data sources using schema integration is involved in constructing multidatabase systems, data warehouses, and integrating data sources within organizations and on the Web. Thus, a mechanism for performing schema integration is of great theoretical and practical importance. Automated integration procedures cannot be applied without a systematic way of capturing the meaning of the stored data. Thus, it has been proposed that automatic schema integration is not possible [8].

Databases have been integrated using various architectures including multidatabases [4, 19], federated databases [22], and mediator/wrapper systems [20, 7, 15, 11, 18, 3, 1, 13]. At some level all these architectures rely on an integrated view for querying. However, no automatic algorithms have been developed to automatically construct the integrated view and resolve the associated naming, structural, and semantic conflicts

between systems. 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. Therefore, 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.

Previous research 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 Sheth [23] or using some form of global dictionary such as work done by Castano [5]. We believe that the best approach involves defining a global dictionary rather than using rules to relate systems. Rules are more subject to schema changes than a global dictionary implementation and grow exponentially as the number of systems increase.

Mediator and wrapper based systems such as Information Manifold [15], Infomaster [11], TSIMMIS [18], and others [1, 7, 3] do not tackle the schema integration problem directly. These systems focus on answering queries across a wide-range of data sources including unstructured data sources such as web pages and text documents. Some of these systems [1] do not attempt to provide an integrated view. Rather, information is integrated based on best-match algorithms or estimated relevance. Systems such as TSIMMIS [18] and Infomaster [11] that do provide integrated views typically construct these integrated views using designer-based approaches. Then, integrated views are mapped using a query language or logical rules into views or queries on the individual data sources. There is limited discussion on how the integrated global views and the associated mappings are actually created. This is probably because the definition of the integrated views and the resolution of conflicts between local and global views are manually resolved by the designers. Once all integration conflicts are resolved and an integrated global view and corresponding mappings to source views are logically encoded, wrapper systems are systematically able to query diverse data sources.

Thus, wrapper and mediator systems do not solve the schema integration problem. These systems solve the interoperability problem by providing mappings from a global view to individual source views and combining query results. To solve the schema integration problem, an automatic algorithm for producing the global view from data source information is necessary. This work proposes an automatic schema integration algorithm based on removing all naming conflicts by utilizing a standard dictionary of terms to describe schema element semantics.

Internet and industrial standards organizations have taken the more pragmatic approach to the integration problem by standardizing the definition, organization, and exchange mechanisms for data in such a way that it can be communicated effectively. Instead of attempting to integrate entire database systems, their goal is to standardize the exchange of data from one system to another. This is a more straightforward problem because it is widely accepted that communications occur using standardized protocols and structures. Instead of determining data semantics for an entire system, it is only necessary to capture data semantics for the data required in the communication and format it using a recognized standard during transmission.

Work on capturing metadata information in industry has resulted in the formation of standardization bodies for exchanging data. An early data exchange standard was Electronic Data Interchange (EDI) which provides a mechanism for communicating business information such as orders and shipments between companies. As the web has

become more prevalent, standards have been proposed for exchanging data using extensable markable language (XML) and standardized XML schemas based on it. Microsoft has lead a consortium promoting BizTalk [9, 10] which allows data to be exchanged between systems using standardized XML schemas. There is also a metadata consortium involving many companies in the database and software communities whose goal 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 1.1 [6] as an emerging standard for specifying and exchanging metadata. Other standardization efforts include the Standard Interchange Language (SIL) [14].

It is important to notice the fundamental differences between the database research approach and the industrial approach. Research algorithms analyze structural and semantic information to resolve schema conflicts. However, only non-automatic algorithms have been developed because resolving all conflicts is extremely difficult. Industrial systems resolve conflicts by accepting standards. These systems standardize the structure, naming, and organization of data communications whether implemented as a formatted EDI document, a BizTalk schema, or a standardized set of XML tags. Integration is achieved by assuming the problem away with standardization.

Our approach is to combine the two techniques. We believe that some level of standardization is required to achieve automatic integration. Specifically, by accepting a standard term dictionary for describing schema element semantics and thus avoiding naming conflicts, the more complex structural and semantic conflicts can be automatically resolved.

3 Integration Components

Before we describe the overall integration architecture, it is necessary to explain the three components required to achieve it: a standardized dictionary of terms, a metadata specification for capturing data semantics, and an integration algorithm for combining metadata specifications into an integrated view. The dictionary provides a set of terms for constructing semantic names describing schema elements. By defining semantic names using a standardized dictionary, we assume away naming conflicts. That is, two schema elements with the same semantic name are assumed to represent identical concepts regardless of their structural organizations. Metadata specifications called X-Specs store schema information in XML documents. A X-Spec also contains mappings from semantic names used in the global view to system names used in the data source. The integration algorithm matches semantic names to produce an integrated view of concepts.

To illustrate the integration algorithm, we will use the following example involving two books databases. The first company, called **Books-for-Less**, stores its book catalog in a database using the structure: *Book(ISBN, Title, Author, Publisher, Price)*. The second company, called **Cheap Books**, stores its book database with the structure: *Book(ISBN, Author_id, Publisher_id, Title, Price, Description)*, *Author(id,name)*, and *Publisher(id,name)*. Throughout the text, database field and table names will appear in *italics* and their associated semantic names in the global view will be in `true-type`.

3.1 A Standardized 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 standardized 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, simply using an on-line English dictionary for the computer to consult is not practical. The size of the database is a problem, and it is complicated for the computer to determine when two words represent semantically equivalent data.

We define a standardized dictionary of terms which are used to construct semantic names for schema elements. The dictionary itself is defined and transmitted using XML. The dictionary is not a standardized schema which forces all data to be represented in one form. Rather, it consists of a set of terms which represent concepts. These terms are combined into a semantic name for a schema element and then stored in the X-Spec to provide a mapping from the global view to the local view.

The standard dictionary is organized as a tree of concepts. All concepts are placed into the tree and are related using two types of relationships: 'IS-A' relationships and 'HAS-A' relationships. IS-A relationships are the standard subclass/superclass type of relationships and are used to model generalization/specialization data concepts. For example, consider two dictionary terms "customer" and "claimant". In the dictionary, "claimant" would be a subclass of "customer" as presumably a claimant is a special type of customer. 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 first, last, and full name components.

Although this is a rather simple modeling mechanism, it is an adequate for the dictionary. In extensive real-world testing, we have encountered no concept terms that cannot be inserted properly into the dictionary using this procedure. Note that the dictionary is not the integrated view. It is just a standard set of terms to consult to describe semantics to create the integrated view. By analogy, the dictionary is like an English dictionary. It defines the individual semantics of accepted words used to convey knowledge. However, overall semantics are communicated by organizing words into a structure (sentences). Our structure for semantic communication is a *semantic name* whose simplified structure is easily parsed unlike natural language.

3.1.1 Defining the Dictionary

The structure of the dictionary is modeled after the top-level ontological categories proposed by Sowa [24]. We have built a dictionary of terms starting from these top-level ontological categories [16]. For the purpose of this paper, the terms required in the dictionary are: `Book`, `Author`, `Publisher`, `ISBN`, `Title`, `Price`, `Id`, `Description`, and `Name`. The exact placement of these terms in the dictionary is irrelevant for this

discussion and has no affect on the validity of the integration algorithm itself. The exact terms used to convey the semantics and their organization is irrelevant. Different terms can be chosen to represent these concepts and different languages used. The sole requirement of the dictionary is to provide standardized names for concepts with unambiguous definitions.

Our initial dictionary contains a limited set of concepts commonly stored in databases. Obviously, the definition of a dictionary storing all concepts is very difficult and time-consuming. Although we constantly expand the dictionary, it is not feasible to assume all concepts will be present as new types of data and very specialized data would not originally appear in the dictionary. 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 which are not in the standardized global dictionary. These additional links are stored and transmitted along with the metadata information during integration. We expect that the evolution of the dictionary would be directed by some standardization organization to insure that new concepts are integrated properly over time.

It is important to realize that the exact terms and organization of the standard dictionary is irrelevant. Although this may seem surprising, consider that all language is simply a standard for expressing semantics. There is no fundamental reason why the word "apple" should describe an apple. Conceivably, the word "orange" could be used or "X123" as long as that was the accepted terminology to represent the concept of an apple. Similarly, the exact organization of the concept hierarchy and the terms used to represent concepts is irrelevant as long as they are agreed upon. For example, the Standard Interchange Language (SIL) uses standardized field names like F01, F02, and F03 to represent data elements. However, the goal of XML is to be human readable, so the dictionary terms should be recognized English words for their concepts, and the base hierarchy should be evolved in a way that models current standardization efforts and real-world organizations. With this idea in mind, we will not delve into the exact definition of the standardized dictionary because it does not affect the correctness of the approach. Any standardized dictionary can be used as long as it is formatted correctly and has the necessary terms to capture the semantics of every data element to be integrated in the corresponding data sources.

By using English words instead of abstract field names, not only is the metadata specification more readable, but it is easier for the designer to assign correct terms to represent data semantics. However, a word may have a slightly different semantic connotation to different people which may affect their choice of a "correct" term. For the purpose of this discussion, we will assume that a designer correctly associates the proper dictionary term to represent the semantics of an element. For example, consider the case where two databases contain home phone numbers. One designer correctly uses the term "Home Phone#" from the dictionary where another uses the more general term "Phone#". Although integration would still occur correctly, the second designer lost some semantics in his choice of term and technically should have chosen the other term. We will assume that these mis-naming problems are handled using an external error-checking mechanism.

Type	Semantic Name	System Name
Table	[Book]	Book
Field	[Book] ISBN	ISBN
Field	[Book] Title	Title
Field	[Book] Price	Price
Field	[Book;Author] Name	Author
Field	[Book;Publisher] Name	Publisher

Figure 1: Cheap Books Database Schema

3.1.2 Constructing Semantic Names

The definition of a semantic name for a given schema element is not a straight-forward mapping to a single dictionary term because a dictionary term provides only a name and definition for a given concept without providing the necessary context to frame the concept. In BizTalk schemas, EDI, and other approaches, the required context is assumed because a standard schema only applies to a very limited communication domain. For example, there are separate schemas for a purchase order, a shipment notification, and order receipt confirmations. Hence, by defining separate schemas, forcing the communication software to choose a specific schema, and defining separate terms for each schema, a single dictionary term provided both context and concept information. For our system, it is necessary to combine dictionary terms to provide both context and concept information.

A *semantic name* captures the system-independent semantics of a schema element including contextual information by combining one or more dictionary terms. A semantic name has the form:

$semantic\_name =$ "[" CT [[;CT] | [,CT]] "]" [CN]

$CT =< dictionary\_term >$, $CN =< dictionary\_term >$

That is, a semantic name consists of an ordered set of context terms (CT) separated by either a comma or a semi-colon, and an optional concept name term (CN). Each context and concept term is a single term from the standardized dictionary. The comma between terms A and B (A,B) represents that term B is a subtype of term A. A semi-colon between terms A and B (A;B) means that term A HAS-A term B, or term B represents a concept that is part of term A. The context terms provide a context framework for the concept that describes them. Every semantic name has at least one context term. The concept name is a single, atomic term describing the lowest level semantics. Fields have concept names to represent their base meaning void of any context information.

Abstractly, a semantic name is a hierarchy of concepts related by IS-A and HAS-A relationships. Typically in relational databases all terms in a semantic name are related by HAS-A associations. After numerous integrations on real-world databases, we have not encountered a schema element whose semantics could not be adequately captured by combining terms from the dictionary in this manner. For example, the semantic names for the schema elements in the Cheap Books database are given in Figure 1 and for the Books-for-Less database in Figure 2.

In summary, the standard dictionary is a tree of concepts related by either IS-A or HAS-A links and stored in XML format. We have defined a basis standard dictionary but allow organizations to add terms as required. Unlike a standardized schema or simple set of XML tags, the dictionary terms are not used independently to define data

Type	Semantic Name	System Name
Table	[Book]	Book
Field	[Book] ISBN	ISBN
Field	[Book] Title	Title
Field	[Book] Price	Price
Field	[Book] Description	Description
Field	[Book;Author] Id	Author_id
Field	[Book;Publisher] Id	Publisher_id
Table	[Author]	Author
Field	[Author] Id	Id
Field	[Author] Name	Name
Table	[Publisher]	Publisher
Field	[Publisher] Id	Id
Field	[Publisher] Name	Name

Figure 2: Books-for-Less Database Schema

semantics. Rather, terms are combined to form semantic names to describe schema elements.

3.2 X-Spec - A Metadata specification language

The definition of a standardized dictionary by itself is not enough to achieve integration because the dictionary is not defining a standard schema for communication; it is simply defining terms used to represent concepts. These concepts can be represented in vastly different ways in various data sources, and we are not assuming a standardized representation and organization for a given concept. Thus, a system for describing the schema of a data source using dictionary terms and additional metadata must be defined. Our integration language uses a structure called an X-Spec to store semantic metadata on a data source. The X-Spec is essentially a database schema encoded in XML format. The database schema is organized in relational form with tables and fields as basic elements.

A X-Spec consists of the relational database schema being described along with additional information about keys, relationships, and field semantics. More importantly, each table and field in the X-Spec has an associated semantic name built from terms in the standardized dictionary as previously discussed.

Describing a data source using a X-Spec looks very similar to a standardized schema developed in BizTalk. We have made an attempt to follow emerging industry standards in the description of schemas using XML and model a X-Spec schema description after BizTalk schemas. The important distinction between a X-Spec schema and a BizTalk schema is that an entire BizTalk schema is standardized. A X-Spec describes a database dependent schema rather than conform to one. As a X-Spec is intended to capture as much metadata as possible about a database schema, additional XML tags are used in its specification that are not present in BizTalk schemas.

A X-Spec is constructed using a specification editor. We have built a prototype version of a specification editor whose purpose is to parse existing relational schema and format the schema information into a X-Spec. Then, the software allows the user to modify the X-Spec to include information that may not be electronically stored such as relationships, foreign key constraints, categorization fields and values, and other undocumented data relationships. More importantly, the specification editor allows the user to construct a semantic name for each schema element.

The use of XML for describing an X-Spec is not required, but it is used because

XML is an emerging standard to exchange semantics between systems. However, the definition and usefulness of a X-Spec is not tied to XML. Information stored in XML in a X-Spec can just as easily be transmitted as formatted text files or structured binary files. XML is used for convenience and interoperability with emerging standards on semantic exchange.

In summary, a X-Spec is database schema and metadata information on a given data source encoded in XML. Although we use XML, a X-Spec is basically structured schema metadata that can be exchanged between systems. A X-Spec is different than a BizTalk schema because it is not intended as a standardized schema for data communication. Rather, it is a document describing an existing database schema which stores semantic names to describe schema elements. As such, there will be a different X-Spec for each data source.

3.3 Integration Algorithm

The integration algorithm is a straightforward matching algorithm of terms. The same term used in two different X-Specs is assumed to represent the identical concept regardless of its representation. The algorithm receives as input one or more X-Specs describing database schema. It then uses the semantic names present in the X-Specs to match related concepts.

Accepting a linguistic framework to prevent naming conflicts is required for integration, but imposing a structural organization to concepts is not necessary. Our architecture identifies similar concepts by name regardless of their physical or logical representations in the individual data sources. The integration result is a hierarchy of contexts and concepts which implies no particular physical representation. The physical representation of the concepts is irrelevant to the user. Users accesses data sources through semantic names which map to physical schema elements. Thus, by not imposing structural constraints on concept representation, knowledge from systems is combined regardless of data representation characteristics, and the user is provided with only the relevant information.

In summary, our integration is valid because it correctly combines database schema into an integrated view given the assumption of no naming conflicts. The architecture avoids naming conflicts by developing and using a standard dictionary of terms and combining them appropriately into context and concept information to express schema element semantics. Since the semantic names constructed are assumed to represent the same concept if their names match, integration of concepts across schema is possible simply by matching semantic names. Concepts are correctly integrated across data sources solely by name regardless of their implementation or physical structure. This allows identical concepts to be combined into an integrated view regardless of their representation and isolates the user from the complexities of data distribution, organization, structure, and local naming conventions.

The integration algorithm is given in Figure 3 and the integrated view produced by combining the X-Specs for the Cheap Books and Books-for-Less databases is in Figure 4.

Given this conceptual view, a user issues queries simply by choosing which fields should be displayed in the final result. The required joins between the tables are automatically inserted by the query processor. The order in which X-Specs are integrated is irrelevant, and the same X-Specs can be integrated several times with no change. As

```
Proc Integrate(X as X_Spec, ByRef V as View)

    For each semantic_name SN in X                                          (1)
        For each term T of SN                                               (2)
            If T does not match any term at this depth Then                 (3)
                Add all remaining terms of T to V                           (4)
                Exit For Loop                                               (5)
            Else                                                            (6)
                Current term = matching term in V                           (7)
                Increase depth in V by matching from current term           (8)
            EndIf                                                           (9)
        Next                                                                (10)
        Add mapping from full SN to schema element in V                     (11)
    Next                                                                    (12)
End Proc
```

Figure 3: Integration Algorithm

V (view root)
- [Book]
  - ISBN
  - Title
  - Price
  - Description
  - [Author]
    - Id
  - [Publisher]
    - Id
- [Author]
  - Id
  - Name
- [Publisher]
  - Id
  - Name

Figure 4: Integrated View

more X-Specs are integrated into V, the number of fields and concepts would grow, but assuming the semantic names are properly assigned, the integration procedure would be unchanged.

4 The Integration Architecture

The three components described in the previous section: a standardized dictionary of terms, X-Specs for storing database schema, and an integration algorithm combine to form the basis of our integration architecture. The integration architecture actually consists of two separate and distinct phases: the *capture process* and the *integration process*.

In the capture process, the X-Spec for a given data source is constructed, and the semantics of the data elements are mapped to a semantic name. This capture process is performed independently of the capture processes that may be occurring on other data sources because the only "binding" between individual capture processes at different data sources is the use of the dictionary to provide standardized terms for referencing data. The specification editor tool is used to extract database schema, add the semantic names and additional metadata, and store the result in a X-Spec.

The integration process actually performs the integration of various data sources. For the purpose of this discussion, it is assumed that there is a central site where the integration is performed by combining the X-Specs of the data sources. Clients wishing to access the individual data sources submit their transactions to this central site which handles the necessary mappings and transaction management. Distributed integration and transaction management where there is no central integration site is an area of future work.

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 is not affected by the capture process at any other data source. This allows the capture process to be performed only once, regardless on how many data sources may actually be 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 central site takes the X-Specs of the individual data sources and executes the integration algorithm to produce an integrated view. This view is then provided to clients for querying. The integrated view displays to the user all elements described using their semantic names and is queried by these semantic names. A query is sent to the central site which performs the necessary mapping from semantic names to system names and divides the query into subqueries against the data sources. The central site is assumed to implement the functionally of a MDBS manager which includes transaction management and query processing. Once results are returned from the individual data sources they are integrated based on the unified view and then returned to the user.

It is important to note that by the use of a central site no translational or wrapper software is required at individual data sources. Once the X-Spec has been provided for the data source and integrated by the central site, the software at the central site communicates directly with the data sources using ODBC or proprietary protocols. All

translation, integration, and global transaction management is handled by software at the central site. This allows full autonomy of the underlying participating databases as the central site appears as just another client issuing transactions to the database.

Our integration architecture contributes several new ideas:

- The definition of a hierarchial, standardized dictionary of terms which can be used across industries and organizations.
- A system for representing in XML entire database schema using the standardized dictionary and metadata about the data sources. (X-Specs)
- An integration algorithm which automatically combines X-Specs into an integrated view allowing global queries.

The importance of the work is the unification of two different approaches to a similar problem. By recognizing the usefulness of industry standards for data exchange, the architecture performs automatic relational schema integration by utilizing a standard dictionary of terms.

5 Querying the Integrated View

The integration architecture automatically produces an integrated view of concepts from the semantic names of schema elements. The integrated view is not a structural view consisting of relations and attributes. Rather, it is a hierarchy of concepts and contexts which map to physical tables and fields in the underlying data sources. Thus, the method for querying the integrated view is different than existing systems, and implementing the query processor results in an entirely new set of challenges.

5.1 Query Formulation and Execution

Users query the integrated view by selecting the semantic names of concepts to include in the result. These semantic names map to physical fields and tables in the underlying data sources. The user is not responsible for determining joins between physical tables in a given data source or across data sources. The system handles the necessary joins based on the relationships between the schema elements. The query system implementation is similar to MIX [2] except that the query is formulated on an integrated view of concepts rather than mediated views.

In many cases, there is a straightforward mapping from semantic names to physical fields. Typically, a semantic name will have only one mapping to a physical field in each data source. Given a list of semantic names in the query used either for projection (inclusion in the result) or for selection criteria, the query processor maps the semantic names to system names using information stored in the X-Spec. To handle joins between tables, X-Specs store information on join conditions between tables for use by the query processor. Thus, all the required mapping information is present to construct a select-project-join query which is translated to SQL.

Currently, the system constructs SQL queries for each data source by mapping semantic names to physical fields and tables. Joins are selected by the system from X-Spec information. If no joins exist between tables, a cross-product is used. Joins across databases are not currently supported as query results from each data source are unioned together. If a given data source does not have all the fields required in the

result, the field is left blank. Obviously, this method of query generation is simplistic. However, many queries can be specified correctly using this mechanism. A more detailed treatment of query issues including join selection and optimization and SQL generation is available [16].

5.2 International Issues

The dictionary of terms is in the English-language. However, there is no fundamental reason for selecting a particular language or set of dictionary terms for use in the architecture. Dictionary terms are effectively place-holders describing semantics, and thus can be arbitrarily assigned as required.

For deployment in international environments, each dictionary term can be associated with an equivalent semantic term in any other language. As long as there exists unique mappings between each dictionary term and a term in another language used to convey the same semantics, non-English language terms can be deployed.

A related issue not covered directly by the architecture relates to data integration. Even though the schema elements may be identical semantically, the actually physical representation in terms of types, sizes, currencies, and scaling factors may be different. There has been work performed on these data integration problems [12, 21]. Note that these problems can be resolved by mapping functions which convert from one context to another. These mapping functions can be automatically applied when sufficient metadata describes the format of the data elements. Thus, this is largely an orthogonal issue to schema integration. For a complete solution, our integration system stores sufficient metadata to automate these mapping functions and format query results appropriately.

6 Architecture Applications

Automatic integration of relational database schemas has numerous applications in the construction of multidatabases, data warehouses, and the interoperability of databases on the World-Wide Web or within an organization. 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 X-spec. The client's browser would connect to the integrated central site to pose queries on data sources that it has integrated. A portal like Yahoo could then combine data sources together as a central site for the user to gather information. Even more exciting would be a distributed version of the architecture, where there is no central site and the user's browser is responsible for the necessary translation and management. When the user wishes to purchase an item, the browser downloads the X-Specs from the on-line stores, integrates them using the

standardized dictionary, and then allows the user to query all databases at once through the "global view of web sites" that was constructed. Obviously, the integration itself is complex, but a system which achieves automatic integration of data sources would have a major impact on how the Web is used and delivered.

It is important to distinguish the integration architecture from wrapper and mediator systems. Mediator systems either assume an integrated view of the data sources is constructed a priori by designers or do not construct an integrated view. If an integrated view is constructed, it is a conventional, structural organization of the data into relations and attributes. This integrated view is then mapped to the local views of the mediators by logical rules or query expressions specified by the designer. Thus, these systems achieve database interoperability by providing an integrated view and its associated mappings to local systems, then automatically process a query specified on the integrated view into queries on the individual data sources. Numerous such systems [7, 15, 11, 18, 3, 1, 13, 20] have been developed.

Mediator systems do not perform schema integration. Schema integration, or the actual construction of the integrated view, is performed manually by designers. Our work is unique in that it automatically produces an integrated view from data source specifications developed independently of other data sources and the global view itself. Thus, the scalability of the system is improved. The integrated view does not display structure and organization to the user. Rather, it hides as much structural information from the user as possible and displays information in a semantically intuitive hierarchy of contexts and concepts. Since the integrated view is no longer queried by structure, new query systems [16] are being developed to compliment the unique nature of the architecture.

We have implemented the integration architecture in a software package called Unity [17]. Unity allows for the construction and modification of the standard dictionary, automatic extraction of metadata into X-Specs, construction of semantic names, and execution of the integration algorithm. We have integrated numerous commercial and research database schemas using Unity. Integrating the Northwind database provided with Microsoft Access with another order-entry database was accomplished in less than a day. We continue to expand the functionality of Unity including refinement of the query processor.

7 Future Work and Conclusions

In this paper, we have detailed how a standardized global dictionary, a formalized method for capturing data semantics (X-Specs), and an integration algorithm can be combined into an integration architecture that integrates relational schemas. Data sources can be indirectly queried by semantic names, and a central site acts as an intermediary providing the necessary integration of concepts and translation between semantic and system names. Integrating entire data sources is a major step forward from industry standards such as EDI or BizTalk which are inflexible and only integrate a small subset of the data actually involved in communications. Also, the system automatically constructs an integrated view as opposed to manual integrated view construction in mediator systems. Applications of our approach include integration of web based databases and integration of entire company database systems allowing easier deployment of advanced technologies such as data warehouses and decision support

systems.

Future work includes refining the standardized dictionary, expanding the functionality of Unity, and developing advanced query algorithms for this unique environment.

References

[1] M. Barja, T. Bratvold, J. Myllymaki, and G. Sonnenberger. Informia: A mediator for integrated access to heterogeneous information sources. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM-98)*, pages 234–241, New York, November 3–7 1998. ACM Press.

[2] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-99)*, pages 597–599, 1999.

[3] S. Bressan, C. H. Goh, K. Fynn, M. Jakobisiak, K. Hussein, H. Kon, T. Lee, S. Madnick, T. Pena, J. Qu, A. Shum, and M. Siegel. The COntext INterchange mediator prototype. *SIGMOD Record*, 26(2):525–527, May 1997.

[4] M.W. Bright, A.R. Hurson, and S.H. Pakzad. A taxonomy and current issues in multidatabase systems. *IEEE Computer*, 25(3):50–60, March 1992.

[5] S. Castano and V. Antonellis. Semantic dictionary design for database interoperability. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pages 43–54, April 1997.

[6] The Metadata coalition. Metadata interchange specification. Technical Report version 1.1, The Metadata coalition, August 1997.

[7] C. Collet, M. Huhns, and W-M. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, December 1991.

[8] B. Convent. Unsolvable problems related to the view integration approach. In *Proceedings of the International Conference on Database Theory*, pages 141–156, September 1986.

[9] Microsoft Corporation. BizTalk Framework 1.0 - Independent Document Specification. Technical report, Microsoft, November 1999.

[10] Microsoft Corporation. Microsoft Biztalk Server - Whitepaper. Technical report, Microsoft, May 1999.

[11] M. Genesereth, A. Keller, and O. Duschka. Infomaster: An information integration system. *SIGMOD Record*, 26(2):539–542, May 1997.

[12] C. Goh, S. Bresson, S. Madnich, and M. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293, July 1999.

[13] J.R. Gruser, L. Raschid, M. Vidal, and L. Bright. Wrapper generation for WEB accessible data sources. In *6th Int. Conf. on Cooperative Information Systems*, pages 14–23, New York, 1998.

[14] Uniform Code Council Inc. SIL - Standard Interchange Language. Technical report, January 1999.

[15] T. Kirk, A. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *AAAI Spring Symposium on Information Gathering*, 1995.

[16] R. Lawrence and K. Barker. Multidatabase querying by context. Technical Report TR-00-16, Department of Computer Science, University of Manitoba, July 2000.

[17] R. Lawrence and K. Barker. Unity - a database integration tool. Technical Report TR-00-17, Department of Computer Science, University of Manitoba, July 2000.

[18] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman, and M. Valiveti. Capability based mediation in TSIMMIS. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 564–566, June 1998.

[19] W. Litwin, L. Mark, and M. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[20] M. Roth and P. Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 266–275, 1997.

[21] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, June 1994.

[22] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogenous and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[23] A. P. Sheth and G. Karabatis. Multidatabase interdependencies in industry. In *Proceedings of the 1993 ACM SIGMOD Conference on Management of data*, pages 483–486, June 1993.

[24] John F. Sowa. Top-level ontological categories. *International Journal of Human-Computer Studies*, 43:669–685, 1995.