The Information Integration Wizard (IWiz) Project

Report on Work in Progress

Joachim Hammer
Dept. of Computer & Information Science & Engineering
University of Florida
Gainesville, FL 32611-6120
U.S.A.
jhammer@cise.ufl.edu

1 Problem Statement

Integration of heterogeneous data sources has been the motivation for research projects throughout many years [2, 3, 6, 10, 11, 21]. The paradigm shift from centralized to client-server and distributed systems, with multiple autonomous sources producing and managing their own data, has created a great demand for data and systems integration. In order to combine information from independently managed data sources, integration systems need to overcome the discrepancies in the way source data is maintained, modeled and queried.

Some aspects of the heterogeneity among data sources are due to the use of different hardware and software platforms to manage distributed databases. The emergence of standard protocols and middleware components, e.g. CORBA, DCOM, ODBC, JDBC, etc., has simplified remote access to many standard source systems possible. Most of the research initiatives for integrating heterogeneous data sources focus on overcoming the schematic and semantic discrepancies that exist among cooperative data sources, assuming they can be reliably and efficiently accessed by so-called *integration systems* using the above protocols. The main problems related to the integration of sources with different schemas and data semantics are the identification and the solution of conflicts between schema and data.

To illustrate the need for integration of heterogeneous data sources, suppose we have a scenario where consumers want to purchase computer-related products from one of the many e-commerce sites. However, before making the purchase they would like to gather all the relevant, available information in order to help them in their decision making process. For example, consumers may want to access product information on available desktops, laptops, software, and other accessories and check availability and pricing information. In addition, users may also want to access other online sources for related background information such as consumer reports, press releases etc. This situation is depicted in Figure 1.

Typically, each source uses different tools and data modeling techniques to create and manage their data. This means, the same concept, for example, the entity `software`, may be described by a different term and different set of attributes in different sources (e.g., `application`, `program`). Also, there is no restriction on the underlying data model used to store and retrieve the source data. Catalogs can be tables in the relational model, persistent objects in object-oriented databases, XML documents published on the web, or flat files kept by legacy systems, to enumerate a few possibilities for data representation. Note, a consequence of the usage of different data models and systems to represent information is the fact that sources provide different query capabilities.

In this online shopping scenario, users often want to issue queries without being aware of which sources are available, where they are located, how they represent their data, and how they are individually queried. Instead, users prefer the illusion as if the data were retrieved from the same repository. Although integration systems do not restrict the way sources create and manage their data, they are built to provide users with transparent, integrated access to a wide variety of heterogeneous data. In order to accomplish this goal, the integration systems must deal with the complex issues related to the discrepancies that result

from the way the data is represented by different sources. For instance, one catalog listing computer books may represent authors' names as single character strings, whereas another source represents them as three distinct strings: first, middle, and last. In another example, the price of a book in one source may include sales tax, whereas another source may separate base price and tax into different attributes.

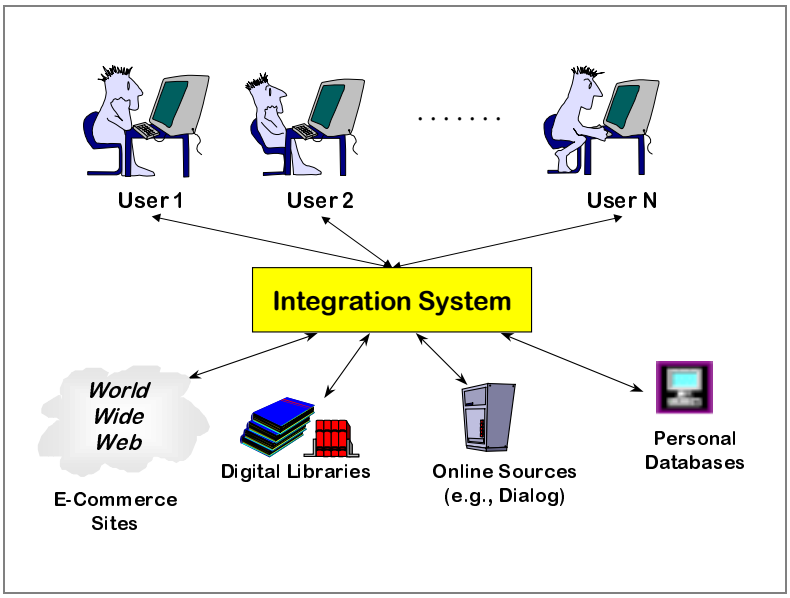

Figure 1: Integrating data from multiple, heterogeneous information sources.

2 Goal of Project

In this research, we will develop and build an integration system using a hybrid data warehousing/mediation approach. The proposed system provides integrated access to heterogeneous data via one common interface and user-definable views over the integrated data. The warehouse component is used to store frequently accessed data for faster retrieval. The mediation component is used to support on-demand querying of the underlying sources in case the desired data is not available in the warehouse. Due to rapid growth of the Web, internet sources containing semi-structured data (e.g., online digital libraries, online portals and bibliographies, e-catalogs, etc.) are of special interest to our project.

To achieve our goal, we have established the following two objectives:

- Develop new algorithms for information integration which are considerably more automated and effective than existing ones. Our focus is on simplifying the tasks of removing structural and semantic discrepancies and merging of related, overlapping data. We propose to carry out this research in the context of the XML representation language [19], which not only fuels the current Web revolution but also opens some exciting possibilities for information sharing. We will use XML together with a suitable, hierarchical data structure (e.g., graph) as our common data representation and model. For example, Goldman et al. Argue in [7] that XML should not be considered a true data model because it specification does not define how to translate an XML document into data structures.
- Demonstrate the viability of our algorithms in a working prototype system which is based on the data warehousing approach to interoperation. We refer to our system as "Integration Wizard" (IWiz) since it lets users access and retrieve information from heterogeneous sources without knowledge about the underlying sources providing the data. Besides representing research in its own right, the goal of this prototype implementation is to build a testbed for analyzing our algorithms and to provide an opportunity for our students to get hands-on experience with a working integration system.

3 Research Issues

During the course of this project, we will investigate the following issues:

3.1 Common Data Model and Representation

One of the most important issues facing the researchers in this project is the choice of data model and representation which will be used by the algorithms and components in IWiz. To date, there are many proposals for data models suitable for representing information in an integrated system: Stanford's Object Exchange Model used in the TSIMMIS project [13], ODMG's object model used in Garlic at IBM Almaden [9], rule- and logic-based systems such as KIF and KQML, and lately XML (eXtensible Markup Language) [4], a proposed document format for exchanging data on the Web. Advantages of XML are its self-describing nature, the clear separation of schema and data, the ability to define new metadata such as data types (Document Type Definition – DTD) and presentation rules (eXtensible Stylesheet Language – XSL [20]), as well as the close connection to the Web and HTML for presentation purposes. In addition, there are already a number of XML parse programs available and the tool support is growing. Although XML has recently only entered its final approval stage, we believe that it will become standard and that industry support for generalized XML and its specific formats will quickly make XML tools as ubiquitous as those for HTML.

However, since XML is only a document format, we also need the ability to map XML-encoded information into a true data model. Given its closeness to the object-oriented model with its ability to store objects persistently, the combination XML (for data formatting) and an object model (for the internal data representation) appears to be promising. Based on our initial investigation, we see the opportunity for XML to make data integration as it has been envisioned in many research projects a reality. Therefore, we have decided to use XML as our common data representation and the Document Object Model (DOM) [15] as our internal data model for storing and manipulating XML data. However, DOM lacks some of the features found in true object-based data models; thus we may replace it with a real OO data model to simplify the integration of an OODBMS as our warehouse.

3.2 Defining the Global Schema

Information sharing using the data warehousing approach begins with an integrated, global schema over the underlying sources where the data originally resides. The purpose of the global schema is to provide a representation of relevant data tailored to the users' needs. Using the global view, users and applications, henceforth referred to as clients, can indirectly formulate queries against the underlying sources without having to understand the actual source schemas.

There are two issues involved in the creation of a global view: First and foremost, the global view should reflect the informational needs of the clients, and should therefore be defined in terms of a global, client-centric view of the data. In addition, the view should be independent of the underlying sources which may be external to the enterprise and out of its immediate control. Consequently, the data in the sources should be defined in terms of the global schema and not the other way around. In other words, data integration should follow the local-as-view approach (as opposed to *global-as-view* approach in which each local source is defined as a "view" of the global schema.). The local-as-view approach simplifies the deletion or addition of new sources as well as updates to existing sources. By definition, the global view and therefore all existing mappings from other sources can remain unchanged; only the mappings from the new source into the global view schema must be generated. As a drawback, the underlying sources may not be complete enough to provide data for every aspect of the global schema. In this case, not all user queries will produce an answer. Thus, care must be taken to ensure that the global view does not contain extraneous information which is typically not found in sources.

Secondly, the global view must be correct and must contain a complete specification of the underlying metadata, including data format information, entity identifiers, etc. Correctness and completeness of the metadata are necessary requirements for correct data integration.

3.3 Semantic Heterogeneities

Information integration and sharing refers to the process of retrieving related information from multiple information contexts and using the combined information in a different context (global view). Before one can share information in this way, several hurdles must be overcome: understanding the meaning of the source data, relating it to the global schema, translating the values from the source context to the target context, and merging related data. All of these hurdles are caused by heterogeneities between source and target, which exist at various levels of abstraction. Primarily, one is faced with heterogeneities at the system level (i.e., differences in the underlying hardware architecture, network protocol, operating system), at the data management level (i.e., differences in the data model, access commands), and at the semantic level (i.e., differences in the way related or similar data is represented in different sources).

It is well-known that most of the progress so far has been made in overcoming heterogeneities at the system and data management level using translators and adapters. In this proposal, we are focusing on overcoming heterogeneities at the semantic level. The goal is to relieve users of the burden of having to understand the information as it is represented in the source and decide how to map it into the target context familiar to them. The simple example shown in Figure 2 outlines some of the issues involved in overcoming semantic heterogeneities.

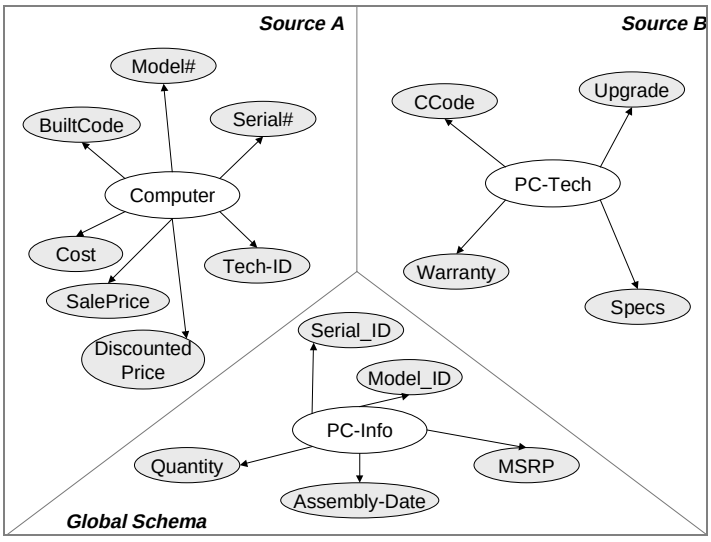

Figure 2: Semantic Heterogeneities.

Continuing the previously introduced computer example, let us assume we have two sources, labeled A and B which contain information on computers and their technical specifications respectively. In addition, we also have a small subset of global schema. For simplicity, let us assume that all three schemas are represented using the same generic object data model: entities are depicted as white ovals, attributes are shaded, and relationships are represented by arrows. Starting with sources A and B, we notice that although the entities have different names, `Computer` in source A and `PC-Tech` in source B, they seem to contain complimentary information. The relationship between these two entities is expressed through matching key values stored in the attributes labeled `Tech-ID` in source A and `CCode` in source B. This allows the user to retrieve, for each computer, the relevant technical specification. On the other hand, the entity `Computer` in source A and `PC-Info` in the global schema appear to represent the same real world concept (based on the similarity of the associated attributes). There does not appear to be a direct association, however, between the entities in source B and the global schema.

When looking at the three schemas, we notice the following heterogeneities: Attributes with different names seem to represent the same data, e.g., `Serial#` and `Model#` in source A and `Serial_ID` and `Model_ID` in the global schema. Furthermore, while source A represents price

information using three different attributes (`Cost`, `Profit`, `Discount`), the global schema only provides one price attribute, namely the recommended retail price or `MFRP`. Finally, the assembly dates for computers are represented differently: as a date code in source A (`BuiltCode`) and as a calendar date in the global schema (`Assembly-Date`). As a result, if we wanted to restructure and merge sources A and B into the global schema, we not only have to figure out which entities and properties can be merged, which cannot, and which are duplicates, but also what transformations to apply to the data before the values mean the same. Besides those semantic heterogeneities shown in Figure 2, there are many others that occur with the same frequencies: different data types (integer vs. string), arithmetic unit conversions (Celsius/Fahrenheit), accuracy conflicts, etc. As part of our research, we are generating an exhaustive classification of all semantic heterogeneities; given our commitment to XML as our internal model, we are specifically focusing on those hetrogeneities that are possible in semistructured data sources. We believe that this topic is worth investigating since there important differences between the number and types of hetrogeneities that can occur among relational schemas (which have received significant attention in the literature) and those that occur in schemas represented by semistructured data models (which have not previously been examined).

3.4 Information Transformations

In order to overcome the semantic heterogeneities, one has to create mappings that transform the data from its source specific representation into the global view. There is also a different set of mappings that converts requests against the global schema into equivalent requests understood by the underlying sources. There are two distinct steps involved in generating the mappings. Each process takes as input the metadata of the previous step as well as a process specification with instructions for how to generate the mappings. As output, each process generates a set of mappings that describe how the input schema is mapped into the output schema.

The first process, called *schema restructuring,* eliminates syntactic as well as semantic inconsistencies in the source schema with respect to the global schema. In addition, the semantics of the information that is contained in the source data model is made explicit and unit conversions are performed (using units of measurement from the integrated view). We refer to this semantically "aligned" schema the *enhanced information source context*. At this point, all schemas from participating sources are using the terminology and relationships specified in the global view.

The second process is called *schema merging* and combines the aligned schemas from the previous step into the integrated view schema defined in the beginning. Activities during the schema merging include removal of duplicates, filling in of missing information, removal of inconsistent data, etc. Using the generated mappings, it is possible to create a program that can transform data from one or more source contexts into an integrated target context. We make use of information transformations in our integration algorithms.

3.5 Knowledge Representation

In order to create a basis for reasoning about the meaning of and relationships among concepts, we need a common metadata knowledgebase that is generally understood which will provides a standard for the meaning of concepts. Such a knowledgebase is called ontology [8]. Ontologies are application-specific and dynamic: Application-specific because each contains most of the commonly used terms and their definitions for a particular domain. Dynamic because the knowledgebase can grow as the need to contain new and updated terms arises. Ontologies have existed for many years and rules and guidelines have been worked out for their standardization. In addition, there exist many support tools for creating, editing, and sharing ontologies (e.g., ontology editors [5]). Ontologies are based on first order logic, making them powerful vehicles for reasoning about relationships. Not all definitions have to reside in a single ontology since there are various algebras for manipulating the contents of ontologies (see work on ontology algebras [18]). The Stanford Knowledge Sharing Group is an excellent source for comprehensive ontologies for many different applications. We envision that sharing ontologies will some day be as commonplace as exchanging documents or e-mail.

4 Approach

Before we describe our approach for restructuring and merging data in the next section, we briefly highlight the proposed IWiz architecture.

4.1 IWiz Architecture

IWiz is based on the data warehousing approach to information integration meaning that data is extracted from the sources, integrated, and cleaned before the users can access the data. Our goal is to build a system that allows its users to design and generate a global schema for a given application environment and then have the system populate the schema with relevant data from one or more sources using integration rules which have been generated beforehand. The global schema as well as the data are stored in a warehouse for efficient browsing and querying. It is our view that a system which supports the generation and materialization of a queriable, global view quickly and efficiently is extremely useful, especially in situations where interesting data is found in sources that do not support a query interface (e.g., Web sites, electronic catalogs).

Of course, not all of the integration steps can be completely automated. Specifically, in IWiz user input is necessary during the following tasks: (1) To specify the semantics of the global view using an ontology. (2) To verify and update the restructuring specifications which are automatically generated by the restructuring algorithm. (3) To verify and update the merging specification generated by the merging algorithm.

An overview of our proposed IWiz integration systems is shown in Figure 3.

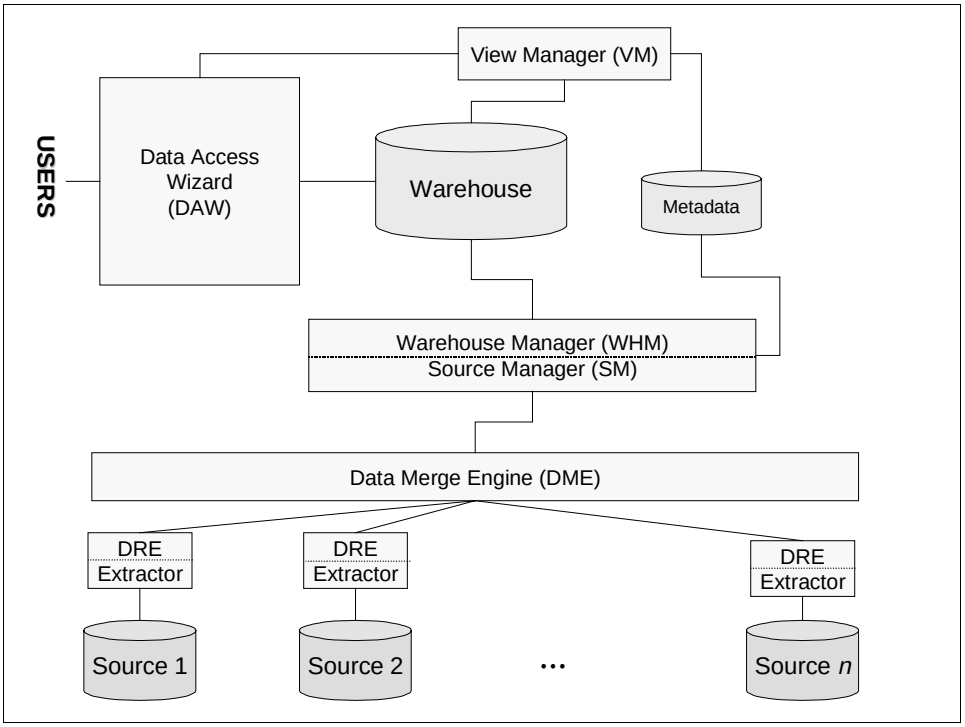

Figure 3: IWiz Architecture Components.

In this figure, we can distinguish seven major components whose functionalities and interactions are briefly highlighted below. A detailed description for each component including design and implementation is given in the corresponding sections later in the document (to be provided during the course of this project).

Starting form the bottom of the figure, we can identify:

1. The data sources. Sources are semistructured, e.g., Web-based; in the beginning we assume that all data is represented using XML, i.e., each source is represented as one or more well-formed XML documents with corresponding DTDs. Note, we are also investigating issues related to the semi-automatic generation of DTD's from XML documents in the cases where the sources have no explicitly defined DTD's.
2. The data restructuring engine (DRE). Its task is to transform the structure and syntax of the XML documents representing the sources into semantically equivalent XML documents that conform to the structure outlined in the target schema in the data warehouse. In subsequent versions of this architecture, we may need to add extractor and monitor functionality to this component to be able to use sources which are not represented in XML.
3. The data merge engine (DME) which fuses the restructured data from the individual sources (i.e., the restructured XML documents) into one representation described by the global target schema. An important part of this fusion step is the reconciliation of conflicts that typically exist in overlapping data coming from multiple sources.
4. The source manager (SM) and warehouse manager (WHM). Both modules interact closely with each other and may be implemented as one component. Initially, the source manager may not be necessary. At a later stage, the source manager may evolve into mediator which processes queries against the multiple sources.
   - The task of the source manager is to maintain a catalog of sources including a description of the data that is stored in each source, query capabilities, schema information (e.g., DTD), and other information that is useful and necessary for locating sources that can contribute data to the target schema. It uses the metadata repository for making source info persistent.
   - The task of the warehouse manager is to maintain the contents of each global schema stored in the warehouse. It is thereby responsible for incorporating the restructured and fused XM document (XML-Doc^M) into the warehouse. Since the warehouse may contain data belonging to multiple applications, the warehouse manager must keep track of the warehouse contents for loading and maintenance. This is done either by overriding the existing contents (easiest but inefficient), appending the data in the new XML document to the existing data in the warehouse (not always possible) or merging the two (hardest). Initially we plan on implementing the simple "complete-refresh" loading approach. Thus maintenance of the warehouse is reduced to periodically refreshing the contents of the relevant warehouse data with a new XML-Doc^M; in later stages of the project, we will implement the sophisticated maintenance and loading operations such as append and merge loading combined with incremental warehouse maintenance, for example.

   The warehouse manager interacts with the source manager to keep track of and communicate with the sources and their associated DRE's that provide the data for a particular database in the warehouse. Sample interactions are

   - Forwarding of the global DTD to the DRE (in case of new source or existing source underwent schema change)
   - Sending maintenance queries to the sources (incremental warehouse maintenance, e.g., [22]
   - Notifications of changes to data and/or schema in one of the sources (either by DRE or source itself).

   In the future, the source manager will evolve into the mediator component [16, 17] of the virtual warehouse systems such as TSIMMIS [2], MIX [1], etc. allowing users to query sources directly. This will require more sophisticated query processing capabilities inside the mediator (e.g., see [14]).
5. The data warehouse (DWH). Stores the data for the various target schemas (one per application domain), the ontologies for describing the terms in each schema, the data for the user views (if

they are materialized) as well as the as all of the metadata for the views, the target schemas, the as well as the sources.

Our main requirement for the data warehouse is persistent support for XML as well as efficient query processing. We have two options: (1) use one of a relational DBMS with our own XML support on top. (2) use a persistent store that supports XML natively. There are several repository systems which support XML natively. Most prominently the LORE system [7, 12] and GMD's XQL query engine with persistence. We are currently in the process of examining and analyzing the pros and cons of both options.

In addition, the warehouse also serves as a persistent repository for storing the conversion specifications as well as several other auxiliary data structures used by both processes such as the thesaurus, source table info, etc.

6. The view manager (VM). The task of the view manager is to create and manage the user views.
7. The data access wizard (DAW). Lets users view data in an easy-to-understand and easy-to-explore fashion using a Web-based graphical interface; rather then writing queries, the user can view and manipulate the data in a query-independent, drag-and-drop fashion. Data manipulation includes the browsing of a global (target) schema for each supported application domain (e.g., health care, computer online help, scientific publications, etc.), the browsing of the corresponding ontology defining the concepts and terms used in the schema, the definition of views which are based on the global schema, as well as the browsing and querying of user views.

Based on our proposed IWiz architecture, we distinguish between two different phases: a built-time phase for (semi-)automatically generating restructuring and integration rules, and a run-time phase for moving data form the sources into the warehouse to support querying and browsing.

4.2 Restructuring and Merging – Build-time

The goal of the restructuring and merging phase (RM) is two-fold:

1. To generate two different sets of rules for converting source data from its native representation into the integrated format specified by the global schema: (1) **for each source** a source-specific restructuring specification for carrying out the schema alignment, and (2) a **single** integration specification for carrying out the merging. The rule sets for both activities are generated in two subsequent steps and will be used as inputs to the restructuring engine and data merger respectively of the run-time phase (Sec. 4.3).
2. To populate the global target schema with the relevant data. This second step is what makes our approach similar to the warehousing approach with all the advantages and disadvantages. It is currently necessary since the merging of data is a rather time and resource consuming activity that cannot be done quickly and efficiently at run-time. It is our goal to avoid this second step altogether and speed up the merging so that is can done during the run-time phase. In that case, we would have a "pre-integrated–data-on-demand" system.

Before we can begin the restructuring and merging of XML sources, we assume that there exists a global schema describing the application area of interest. So, for example, assuming that we are interested in providing access to computer software and hardware, we assume that there is a schema which models computer software and hardware in the most general terms. This global schema is sometimes also referred to as target schema during the restructuring and merging processes.

After the creation of the target schema, the warehouse manager will initiate the loading process (built-time activity). (1) It passes the DTD(s) representing the target schema to the source manager. (2) The source manager will determine which source(s) are available and contain data that needs to be fetched (that is relevant to the target schema). For each such source, the source manager invokes the corresponding DRE by passing it the target DTD. (3) Each DRE will determine if there exists a set of restructuring rules for converting the source XML Docs into the format specified by the target DTD or if the existing set needs to be updated due to schema changes at the underlying source. If yes, the

restructuring process is started and the restructured XML Doc^R is returned to the source manager. If not, the rule generation process is started first. (4) As soon as all restructured XML Docs are received by the source manager, the merging process is initiated with the help of the merging engine. After the restructured XML docs have been merged into one single document, it is passed to the warehouse manager for loading into the warehouse.

4.2.1 Source Data Restructuring

Inputs to the first rule-generation step are an XML document and corresponding DTD representing an information source (XML-DOC and DTD), a description of the global schema (DTD*) as well as an ontology (dictionary) describing the concepts and terms in the global schema. Output is a set of rules represented as XSL directives for converting the original source XML document into a new XML document (XML-DOC^R) which corresponds to the DTD* for the global schema.

Once the rules (XSLT rules) are generated, the transformation of data is done by the XSLT processor which uses the previously generated rules to convert its input (an XML document) into a new XML document which adheres to the structure of the integrated schema. In version 2, we envision the restructuring engine as part of the wrapper component which will provide query translation and processing as well as data model conversion capabilities. Currently, we assume that the source data has already been transformed into an XML Document and that there is an explicit DTD describing its schema. Furthermore, we assume that there are no queries, i.e., that the relevant source data is pushed into the warehouse in a bottom-up approach.

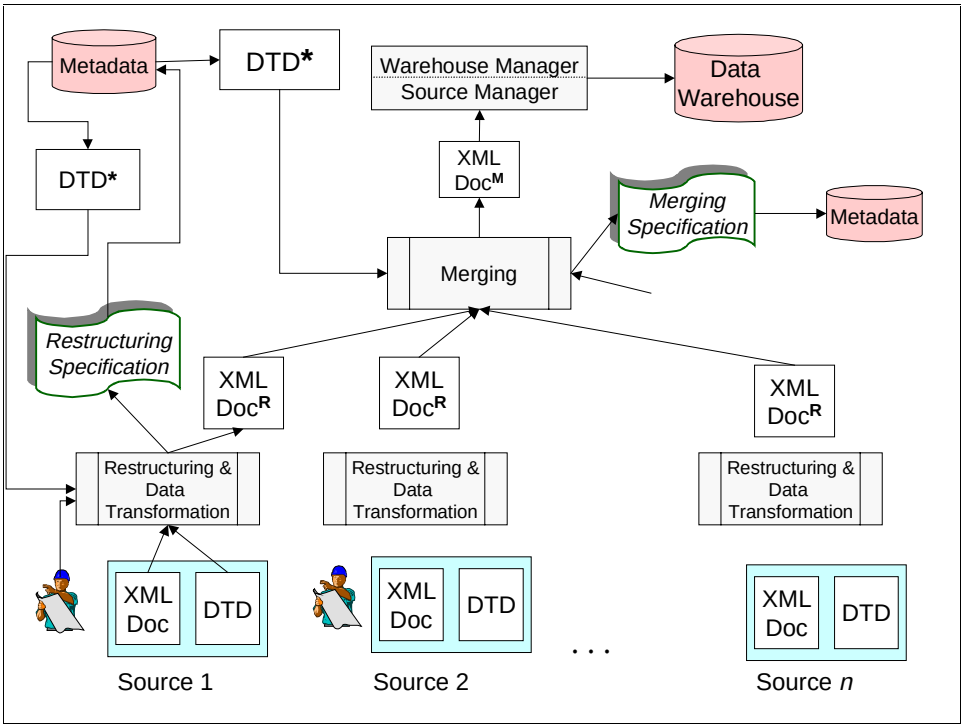

Figure 4: Restructuring and Merging – Built-Time Architecture.

4.2.2 Data Merging

Inputs to the second rule-generation step are the restructured and aligned source data (XML-DOC^R) as well as DTD*. Output is the merging specification represented as a set of XSL directives. Data merging is a 2-step process consisting of a *fusion* step which pulls the data from the individual XML Docs into one followed by a *cleaning* step which reconciles the conflicts caused by the overlap of similar and related data from the various sources.

4.2.3 Build-Time Overview

The interactions between the restructuring and merging processes of this first phase are depicted in Figure 4.

4.3 Data Access – Run-time

The data access phase allows the user to browse and access the integrated data that is stored persistently in the data warehouse. We distinguish two different architecture versions with different components and capabilities. A simple version (Version 1) which provides no query capabilities of source data other than through the warehouse contents, and a sophisticated version (Version 2) which supports distributed query processing through mediators. Our goal is to eventually build Version 2; however, version 1 will serve as an early prototype for evaluating the viability of our key components, namely the restructuring and merging engines, the source, warehouse and view managers, the warehouse, as well as the data access wizard.

4.3.1 Version 1: Warehouse Access with Bottom-Up Information Refresh

The three main components in this phase are the data access wizard, the warehouse and metadata repository. Initially, all data is already stored in the warehouse at the beginning of this phase. If changes to the underlying data sources occur, the warehouse must be refreshed which will require the execution of the steps of the previously discussed built-time phase. Since no maintenance queries are possible in this version, warehouse maintenance is limited to a complete refresh. This extremely simple "maintenance" is done by the warehouse manager. However, since the warehouse can contain multiple schemas, the warehouse manager also needs to keep track of which schema is to be updated.

User queries are only supported in the data warehouse. We envision the use of an XML-based repository such as LORE. Users can query the global warehouse schema directly or through pre-define views. Views and queries are defined and formulated in the Data Access Wizard which provides a graphical interface to the warehouse. The architecture is depicted in Figure 5.

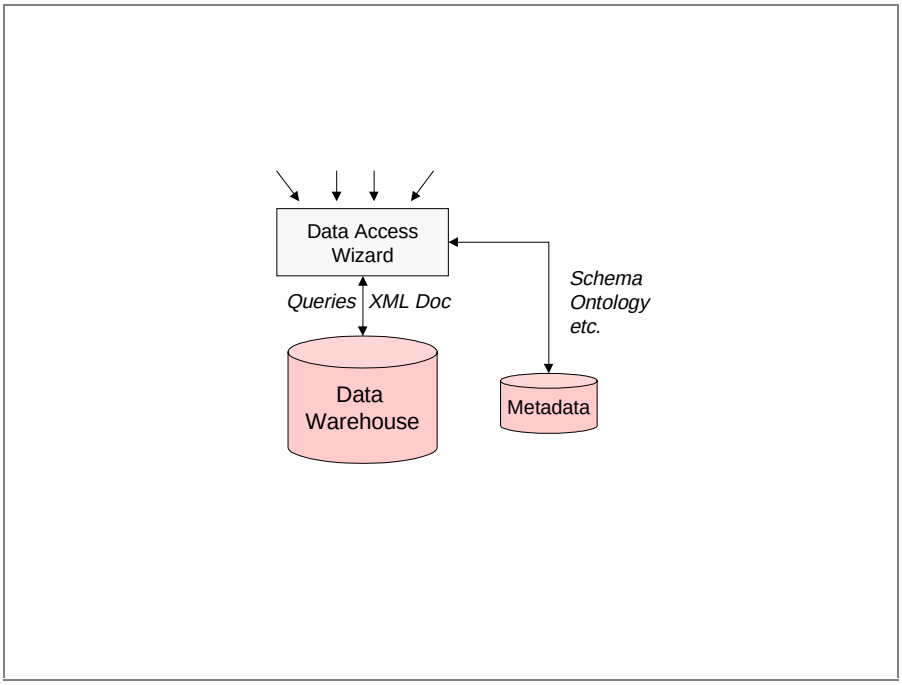

Figure 5: Warehouse Access with Bottom-Up Information Refresh.

Please note that this approach represents a very simplistic data warehouse scenario because we do not even support maintenance queries that help keep the warehouse up-to-date. Rather, we only support

an information push model, where the pre-defined information is "pushed" into the data warehouse at pre-defined times, e.g., when the source data has changed. Thus the name "Warehouse Access with Bottom-Up Information Refresh." Also note, despite its simplicity, there are still many real-world usage cases for this approach.

4.3.2 Version 1.5 (intermediary): On-Demand Warehouse Access with Pre-integrated Sources

The difference between this version and the previous version is that the actual merging of data is delayed until the data is actually requested by the user. However, in order to provide acceptable query response time, the merging process must be separated into two parts just like the restructuring process: generation of the merging specification at built-time using a representative training set of the actual data, and the data merging process at run-time using the merging specification developed before.

At run-time, when the user requests data, the DAW in conjunction with the warehouse manager must determine if the data is already in the warehouse, if it is, the request is serviced just like in the version 1. If not, the warehouse manager, must invoke the DRE and DME to restructure and merge only those source data items which are relevant for the request (alternatively, we might materialize the entire user view for efficiency). In a sense, we have moved the part involving the movement of data from the built-time architecture into the run-time architecture. Thus loading and refresh (maintenance) has now become a run-time activity.

An overview of the architecture is depicted in Figure 6

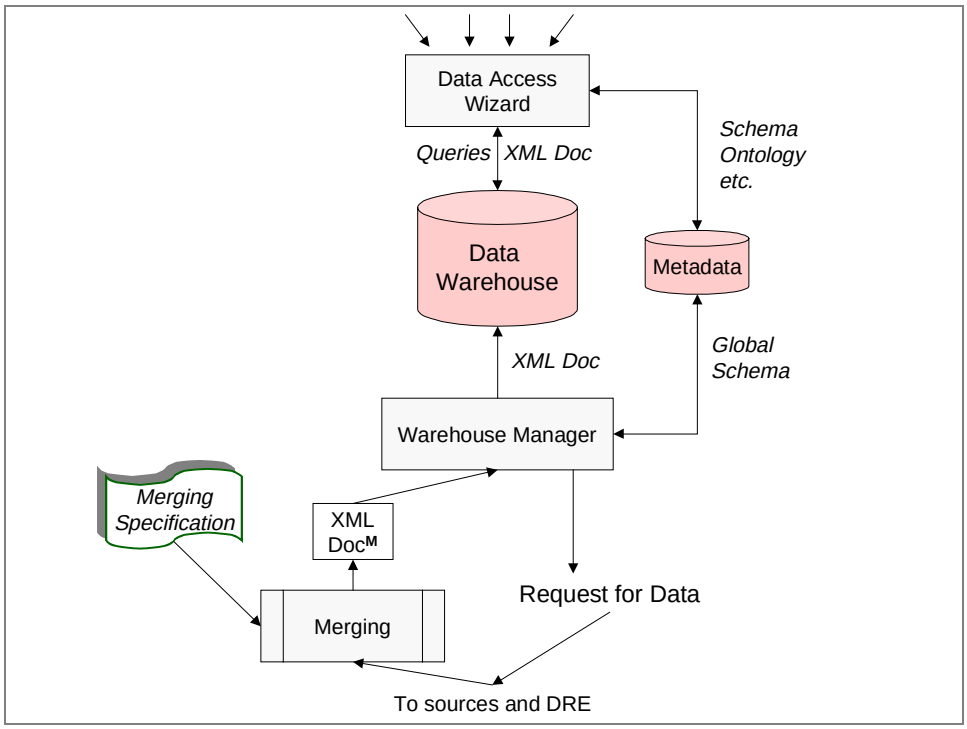

Figure 6: On-Demand Warehouse Access with Pre-integrated Sources.

4.3.3 Version 2: On-Demand Data Access with Warehouse Support

Our goal is to provide an architecture that stores the answers to frequently asked queries in the warehouse (as before); however, it must also allow the querying of sources in case the desired information is not available in the warehouse. Thus in version 2 we will enhance the existing components with distributed querying functionality (= mediation): the integrator will become a mediator (to indicate its distributed query processing power) and instead of restructuring engines we need source-specific wrappers (to indicate their query and data translation capabilities). As before, we may start with scaled-

down versions of both components with more limited capabilities. In this new version, the warehouse manager now serves a dual function, namely the routing of queries to the mediator that cannot be answered in the warehouse as well as the warehouse maintenance in the more traditional warehouse sense (again, this latter capability maybe provided during later stages of the project). In a sense, the we are moving more and more functionality from the built-time architecture into the run-time architecture.

In order to provide mediation capability we need conversion and integration specification mappings that go both ways rather than bottom-up as in the previous version. Thus our main focus here will be on finding ways to enhance our existing mappings (if possible) or to generate mappings for going top-down.

A preliminary overview of the components and their interactions is shown in Figure 7.

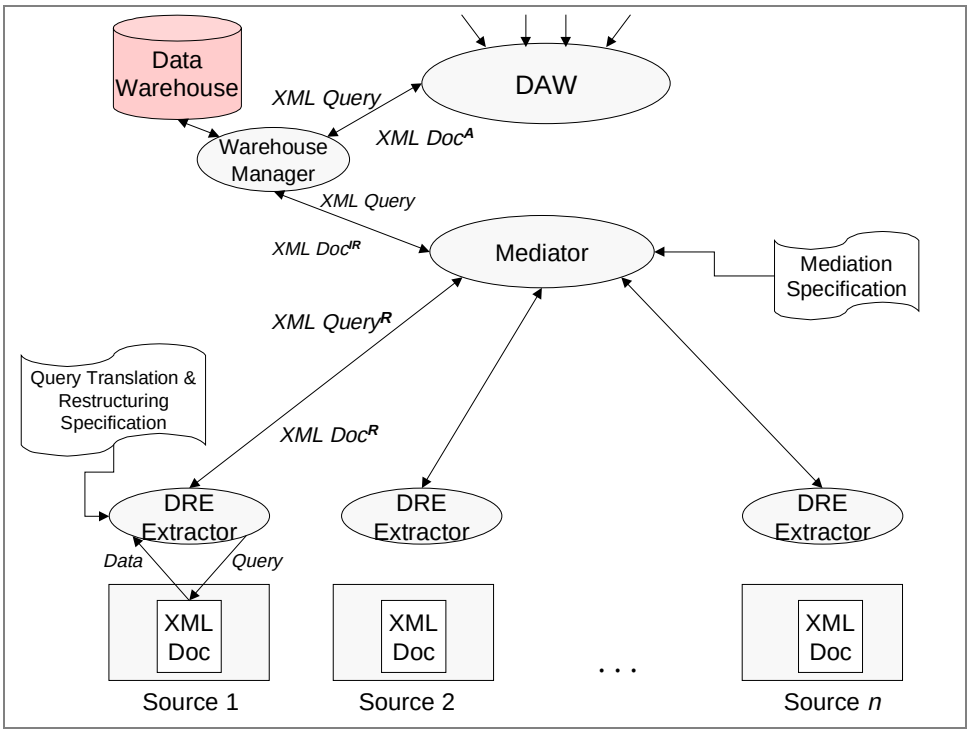

Figure 7: On-Demand Data Access with Warehouse Support.

References

[1] C. Baru, A. Gupta, B. Ludaescher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chen, "XML-Based Information Mediation with MIX," in *Proceedings of the International Conference on Management of Data*, Philadelphia, Pennsylvania, pp. 597-599, 1999.

[2] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "The TSIMMIS Project: Integration of Heterogeneous Information Sources," in *Proceedings of the Tenth Anniversary Meeting of the Information Processing Society of Japan*, Tokyo, Japan, pp. 7-18, 1994.

[3] W. W. Cohen, "The WHIRL Approach to Data Integration," *IEEE Intelligent Systems*, **13**20-24, 1998.

[4] D. Connolly, "Extensible Markup Language (XML)", Web Site, `http://www.w3.org/XML/`.

[5] A. Farquhar, R. Fikes, and J. Rice, "The ontolingua server: A tool for collaborative ontology construction," Knowledge System Laboratory, Stanford University, Stanford, CA, Technical report KSL-96-26, September 1996.

[6] M. R. Genesereth, A. M. Keller, and O. M. Duschka, "Infomaster: An Information Integration System," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, **26**:2, pp. 539--??, 1997.

[7] R. Goldman, J. McHugh, and J. Widom, "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language," in *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, 1999.

[8] T. R. Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition*, **5**:2, pp. 199--220, 1993.

[9] L. Haas, D. Kossman, E. Wimmers, and J. Yang, "Optimizing Queries Across Diverse Data Sources," in *Proceedings of the VLDB*, Bomaby, India, 1997.

[10] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge, "The Stanford Data Warehousing Project," *Bulletin of the Technical Committee on Data Engineering*, **18**:2, pp. 41-48, 1995.

[11] A. Levy, "The Information Manifold Approach to Data Integration," *IEEE Intelligent Systems*, **13**12-16, 1998.

[12] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, **26**:3, pp. 54-66, 1997.

[13] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources," in *Proceedings of the Eleventh International Conference on Data Engineering*, Taipei, Taiwan, pp. 251-260, 1995.

[14] Y. Papakonstantinou and V. Vassalos, "Query Rewriting for Semistructured Data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, pp. 455-466, 1999.

[15] W3 Consortium, "Document Object Model (DOM) Level 1 Specification," W3C, W3C Recommendation, November 1998, `http://www.w3.org/TR/REC-DOM-Level-1`.

[16] G. Wiederhold, "Mediators in the Architecture of Future Information Systems," *IEEE Computer*, **25**:3, pp. 38-49, 1992.

[17] G. Wiederhold, "Intelligent Integration of Information," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, DC, pp. 434-437, 1993.

[18] G. Wiederhold, "Scalable Knowledge Composition (SKC)", URL, `http://www-db.stanford.edu/LIC/SKC.html`, 1997.

[19] World Wide Web Consortium, "Extensible Markup Language (XML)", World Wide Web Site, `http://www.w3.org/XML/`.

[20] World Wide Web Consortium, "Extensible Stylesheet Language (XSL)", World Wide Web Site, `http://www.w3.org/Style/XSL/`.

[21] G. Zhou, R. Hull, R. King, and J.-C. Franchitti, "Data Integration and Warehousing Using H2O," *Bulletin of the Technical Committee on Data Engineering*, **18**:2, pp. 29-40, 1995.

[22] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, **24**:2, pp. 316-27, 1995.