Materializing Web Data for OLAP and DSS

Yan Zhu, Christof Bornhövd, Doris Sautner, and Alejandro P. Buchmann

Department of Computer Science,
Darmstadt University of Technology,
D-64283 Darmstadt, Germany
{zhu, bornhoev, buchmann}@dvs1.informatik.tu-darmstadt.de,
doris.sautner@gmx.de

Abstract. Business decisions must rely not only on company-internal data but also on external data from competitors or relevant events. This information can be obtained from the WWW but must be integrated with the data in a company's data warehouse. In this paper we discuss a system architecture for warehousing Web content for OLAP and DSS. A self-describing object model is used to make the implicit modeling and context assumptions explicit, both for the data obtained from the Web and the data already in the data warehouse. A domain-specific ontology provides a common interpretation basis for data and metadata. We propose an object-relational mapping that takes into consideration the peculiarities of relational data warehouses based on a star schema and propose a mapping rule language to describe the necessary transformation rules. The system framework described in this paper has been implemented in Java.

1 Introduction

Web warehousing is a novel and very active research area, which combines two rapidly developing technologies – data warehousing and Web technology. It provides a suitable approach to systematically discover and acquire strategic information from the Web. This information may be identified, catalogued, managed and then accessed by the end users [22], via search engines or some Web information management system [1, 2, 4, 10, 16]. Web data may be extracted, integrated, represented and materialized in a data warehouse for OLAP and DSS [6, 17, 24], and also may be managed by using a hybrid approach for taking advantage of the data warehousing and the virtual integration approach [3, 21, 23, 26].

External data is important for carrying out meaningful OLAP in the context of e-commerce. For example, an online book shop manager may make use of integrating discount book information or information about new publications from other online book shops in his data warehouse, in order to analyze market trends and make new business plans. As Web technology develops, a huge amount of strategic information is already offered on the Web. Integrating and managing e-commerce data and external data in a data warehouse, implementing OLAP on them, and mining historic data to generate summary information and business rules will benefit business. However, when warehousing Web data we face three main challenges:

- **Data Extraction.** Web sources are dynamic and autonomous. Not only is Web-based data updated frequently but Web sources are dynamically developed as well. Every day new sources are brought on the Internet and already available sources may be changed drastically or even disappear. In addition, providers change the presentation and contents of their sources to react to new developments or to hinder automatic, exhaustive extraction of their data. All this makes determining and automatically extracting relevant data from the Web a difficult task.
- **Preparation/Integration.** E-commerce data comes from daily transaction processing on the Internet, which in most cases is in structured form. External data comes directly from some Web sources and in general is in unstructured or semistructured form. In addition, the sources available online are developed and managed by independent institutions. They represent the same or related information on the basis of different assumptions about their organization and meaning. This is because of different political and cultural contexts, or because of different intentions concerning the use of the data. Therefore, before loading data from different Web sources into the data warehouse, we have to resolve structural and semantic heterogeneity among them and between them and the data already in the warehouse.
- **Representation/Materialization.** The paradigm of the Web is totally different from the paradigm of a data warehouse. Most data in the Web is in semistructured or unstructured form, i.e., is available as HTML or XML data or irregularly structured files, which may change frequently. In contrast, a data warehouse in general is based on a well-structured data model, in most cases a relational data model, and is comparably stable. Thus, we have to map irregularly structured, maybe frequently changing data to a predefined, restrictive database schema.

In order to tackle these challenges, we have proposed a framework for warehousing Web information [26]. We represented Web data using a special data model called MIX (***M**etadata based **I**ntegration model for data **X**-change*)[7, 8]. This model represents data together with a description of their underlying interpretation context and uses domain-specific ontologies in order to enable a semantically correct interpretation of the available data and metadata. Thus, it supports the integration of Web-based information. In our previous work [8, 9] some aspects of Web data preparation/integration are already discussed. In this paper we concentrate on problems of representation/materialization of Web data. The main contributions of this paper are:

- We implemented a system framework and designed a Transformation Processor to accomplish transformation from the object-oriented MIX model to a relational star schema in a data warehouse. In the Transformation Processor, an object/relational mapping approach is adopted. The object identity, complex objects and the multi-valued attribute problem are discussed in this context.
- We defined a Mapping Rule Language to specify the correspondences between MIX objects and tables in the star schema of a given data warehouse.

The rest of the paper is organized as follows: Section 2 gives a motivating example. The system architecture and the mediated data model are introduced in Section 3. Section 4 analyzes the transformation process from Web objects to the star schema of the data warehouse. The mapping rule language is explained in Section 5. Section 6 presents related work. Section 7 provides conclusions and identifies areas of ongoing and future research.

2 A Motivating Example

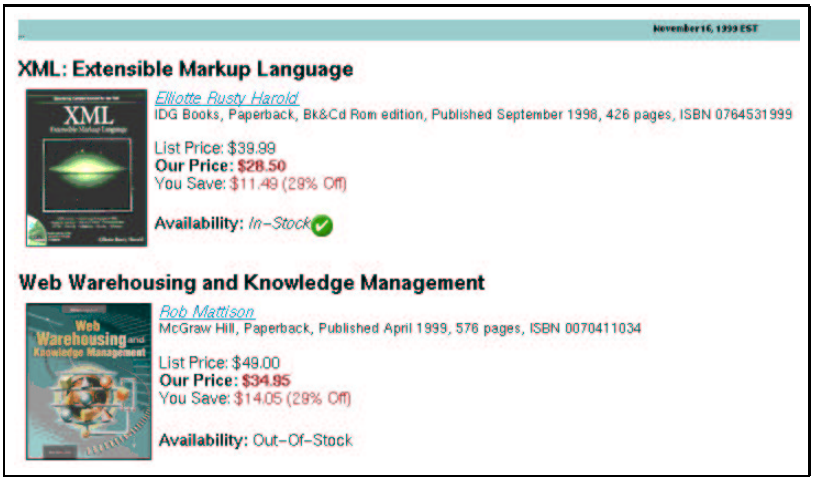

Fig. 1. Discount Books Information from Source A

Fig. 2. Discount Books Information from Source B

Online book shopping is a very active e-commerce area. A large amount of customer orders is produced every day, recorded in a data warehouse for OLAP and decision making. In addition, the book shop manager may also integrate discount book information from his competitors in this data warehouse, in order to compare pricing schemes, analyze market trends and make new business

plans.This information can be obtained from the related Web pages. Figure 1 and 2 show discount book information from two different online providers given as HTML pages. In Figure 3, the star schema of a simplified bookshop data warehouse is defined on the basis of the relational data model.

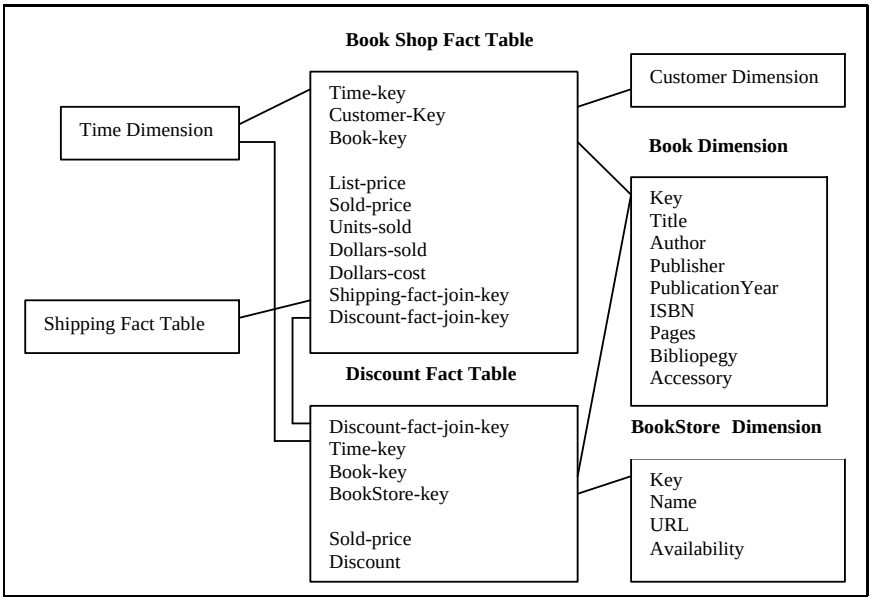

Fig. 3. Star Schema of the Online Bookshop Data Warehouse

3 The Framework for Managing Web Data

We have implemented a Java framework that provides a platform for integrating Web data and mapping it to a data warehouse. The framework, shown in Figure 4, follows the classical mediator approach introduced in [25].

Wrappers are used to extract relevant data from available data sources, to map it to MIX based on a common structural context and to return the MIX objects to the Federation Manager.

The ***Federation Manager*** manages the available data sources by keeping a Metadata Repository. Based on the explicit description of the underlying context the Federation Manager tries to integrate heterogeneous data by converting the data to a common semantic context.

An ***Ontology Server*** stores and manages the domain-specific vocabulary, which in our framework describes a single domain and provides a way to describe the concepts in that ontology independent of any application or data source.

The integrated data is given to the ***Transformation Processor***, which maps the MIX objects to an existing star schema, and loads the data into the data warehouse. It uses the mapping rules specified in the corresponding mapping file. The mapping rules are written using a special Mapping Rule Language (MRL). ***Function rules*** are specified as Java methods stored in the ***Mapping Functions Library*** and can be called by the Transformation Processor. The transformation process is described in more detail in Section 4.

The ***Incremental Maintenance Processor*** serves as the maintenance component of the data warehouse, when Web data is updated. In this processor a copy of the latest MIX objects is kept. The processor reads updated MIX objects, compares them with the old copies and calculates the incremental parts. These are sent to the Transformation Processor, where they are mapped to the star schema of the data warehouse.

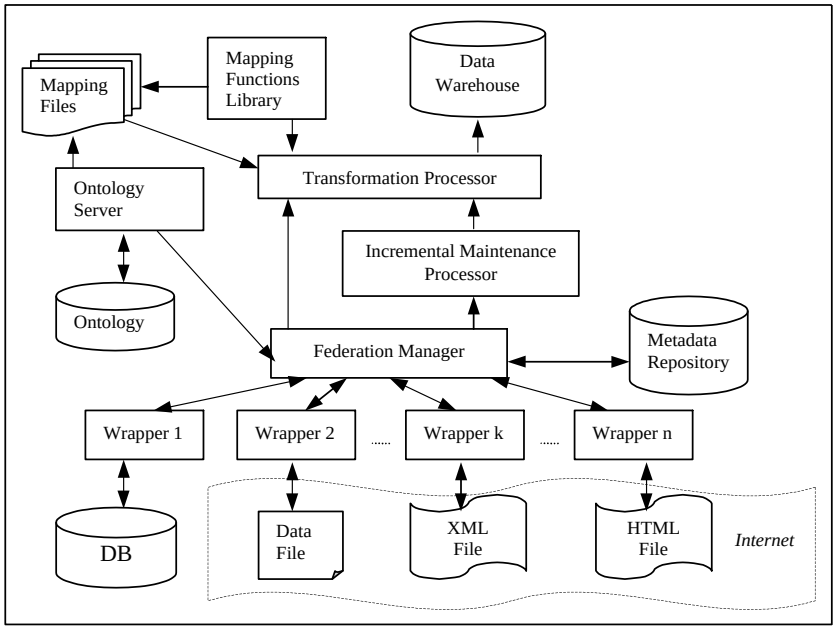

Fig. 4. System Architecture for Warehousing Web Data

MIX is a self-describing data model and is based on the concept of a semantic object. A simple semantic object is represented as a triple of the form: $<ontology\ concept, value, \{semantic\ context\}>$, while a complex semantic object can be represented as: $<ontology\ concept, set\ of\ semantic\ objects>$.

For example, the data given in Figure 1 and 2 can be represented as MIX objects of concept *BookOffer* based on a common structural and semantic context. In our example, we assume dates represented in the form *"Mon DD, YYYY"*, author names given by *"Last Name, First Name Second Name"*, and prices specified in EUR. Figure 5 shows the representation of the integrated data.

4 Transforming MIX Objects to Warehouse Tables

MIX is an object model, while the data warehouse in our framework is based on a star schema. Therefore, mapping MIX objects to the data warehouse tables is comparable to known object/relational mappings as discussed for example in [5, 13, 18–20].

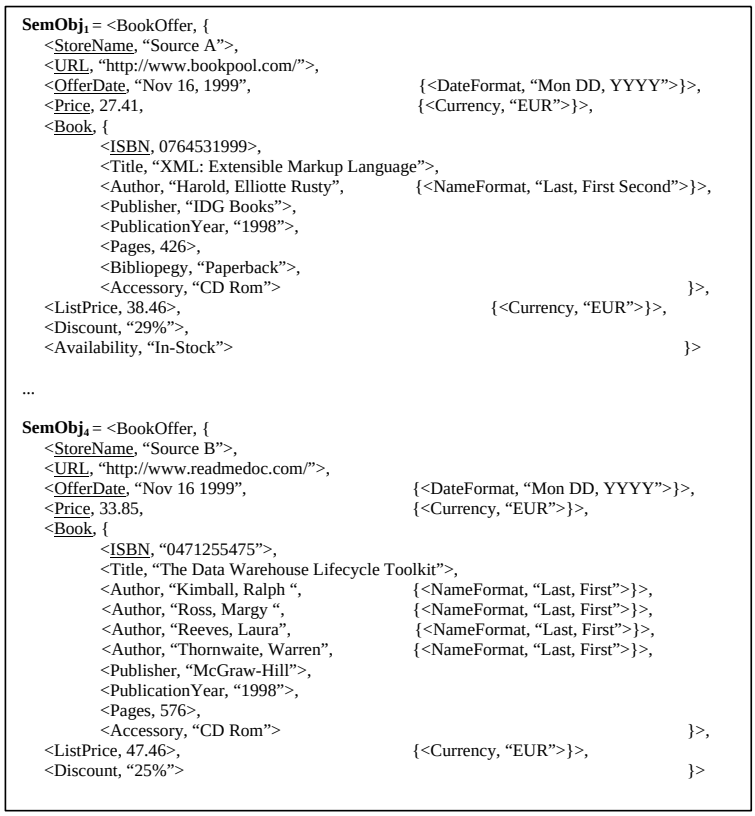

Fig. 5. Integrated Web Data in MIX Representation

However, different to those approaches, data about a real world entity is divided into facts in fact tables and descriptive attributes in dimension tables in a star schema, i.e., is spread across different tables, and the dimension tables must not be normalized. In addition, MIX provides a flexible data model for representing irregularly structured data. Semantic objects of the same concept do not necessarily have the same structure, i.e., set of attributes. For example, in Figure 5 $SemObj_1$ provides information about the availability of a book which is not given in $SemObj_4$. This must be considered when transforming MIX objects into tuples of the star schema. Finally, in contrast to most object/relational mapping approaches which create the relational schema according to the given object classes, we map to an existing data schema in the warehouse.

4.1 Basic Mapping Rules

In our mapping approach we obey the following basic rules:

1. **Aggregations.** In object/relational mapping, aggregations can be implemented using two approaches: single table aggregation or foreign key aggregation [5, 19, 20]. In the data warehousing context we have to distinguish two cases. When aggregation data populates a dimension table, the single table

aggregation approach is favorable because of query execution performance. Thus, we follow this approach. However, its disadvantage is latent long tuple width. The foreign key approach on the other hand might snowflake the data warehouse schema, and lower query performance.
In the case that an aggregate populates a fact table as well as dimension tables, we must decompose an aggregate and separately map subobjects of a MIX object to facts in the fact table and attributes in dimension tables.

2. **Inheritance hierarchy of concepts.** The MIX model supports class inheritance, through the inheritance of concepts in the domain ontology. However, since we have to map MIX objects to an existing data schema we see them only as objects of a concrete concept/class. Therefore, in the mapping we need not consider the inheritance relationships of concepts/classes.

3. **Object/subobject relationships between MIX objects.** In the domain ontology only the relationships between concepts for complex semantic objects and their identifying attributes are specified. Additional, non-identifying attributes may vary between objects of the same ontology concept. However, each MIX object contains concrete attributes. Thus, we have to take possible combinations of attributes of complex semantic objects of a given concept into consideration in our mapping. We achieve this by giving a mapping for all attributes of a class of objects that are of interest for our application.

4. **Handling keys.** In the MIX model each object is identified through single or multipart key attributes, similar to the relational model. When an object is implemented in Java, a unique object ID can be automatically assigned. When a dimensional table is constructed, an additional column can be designed in this table as primary key. One unique key value for each row can be generated by the system. In the star schema of a data warehouse, the primary key of fact tables is the combination of the foreign keys that link to the corresponding dimension tables. The generation of the foreign keys is involved with connecting fact tables and dimension tables. We discuss this rule in 9) in more detail.

5. **One MIX concept to one/many relational attributes.** When mapping an ontology concept for a simple semantic object to table columns we must distinguish three cases. In the easiest case we have a direct 1:1 mapping, i.e., the values of a concept are directly assigned to the corresponding table column. In the second case, we must calculate the values of a column by applying a specified function on the corresponding concept. Finally, when values of an ontology concept must be decomposed to multiple table columns, decomposition functions have to be applied to calculate suitable values.
Calculation rules are specified as Java methods in our framework and stored in the Mapping Function Library. They are called by the Transformation Processor. For example, objects of concept *OfferDate* in Figure 5 specify the date of a given book offer according to the format "Mon DD YYYY". In the Time Dimension of the data warehouse in Figure 3, we have attributes

Day, Month and Year. Therefore, we must use decomposition functions, like: F_{Day} *("Nov 16, 1999")* $\Rightarrow$ *16*, to generate values for these columns.

6. **Many MIX concepts to one relational attribute.** When multiple concepts are mapped to one column, i.e., values from multiple attributes are used to derive a column value, an aggregation function has to be applied to calculate the value. For example, calculating the *TotalCost* paid by a customer in one book purchase is involved with MIX objects of concept *Subtotal* and *ShippingCost*. The corresponding calculation function is given by: $F_{TotalCost}(\ Subtotal, ShippingCost\ ) = Subtotal + ShippingCost \Rightarrow TotalCost$.

7. **Multiple values problem.** When different subobjects of the same concept occur, a multi-valued attribute problem arises. For example, a book may have more than one author. Most relational database systems do not support multi-valued attributes. If the maximal cardinality of such multi-valued attributes is known in advance, multiple columns can be used to store them. Otherwise, separate rows have to be used to store them. Since we map to an existing star schema, we must adapt the mapping to the given table structure. For example, multiple authors of $SemObj_4$ shown in Figure 5, must be mapped to multiple rows to store each author name.

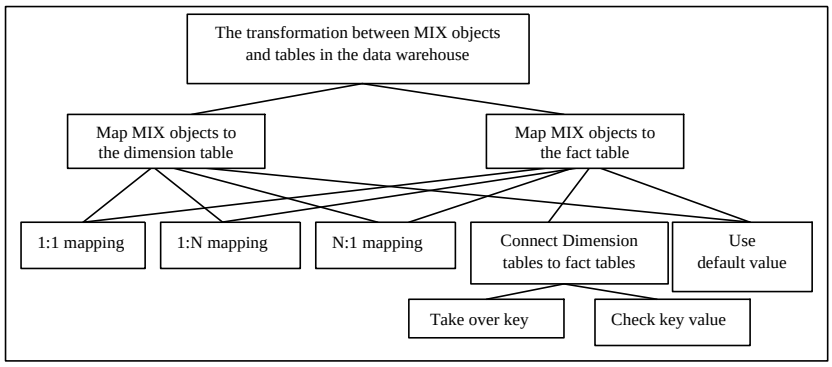

Fig. 6. Several Cases of the Transformation

8. **Default values.** When we map MIX objects to the tables of the data warehouse, we sometimes do not have values for all attributes. It would be necessary to use some default value in these places. For example, $SemObj_4$ shown in Figure 5 has no *Availability* item. In the transformation processing, we will use "Unknown" as the value of *Availability* in a tuple. We can suggest some default values as standard DefaultValue, such as "Null" or "Unknown".

9. **Connecting dimension tables to fact tables.** To connect dimension tables to fact tables, two cases must be considered. In one case, if a new tuple of a dimension table is populated and the primary key value is generated by the system, we can register this value in the corresponding tuple of the fact table as one foreign key value. **Example a** in Section 5 will describe this

transformation. In another case, a dimension table already exists and all key values in this table are already generated before we link dimension tables to a fact table. To connect such dimension tables to the fact table we find the required key value in the dimension table and then take it over in the fact table. **Example b** in Section 5 will explain this situation.

The transformation cases are classified in Figure 6.

4.2 Transformation Processing

In the Transformation Processor shown in Figure 7, the Receiver reads MIX objects from the Federation Manager or from the Incremental Maintenance Processor and puts them in the Vector of MIX objects. The Parser reads mapping rules from mapping files that are written in MRL and parses rule documents. The parsed results are stored as a list in the ListStructure. The Mapper reads mapping rules from the ListStructure and uses mapping functions from the Mapping Function Library. Under the guide of the given rules, the Mapper transforms MIX objects to facts and attributes in the data warehouse. The data is loaded through a JDBC driver into the data warehouse.

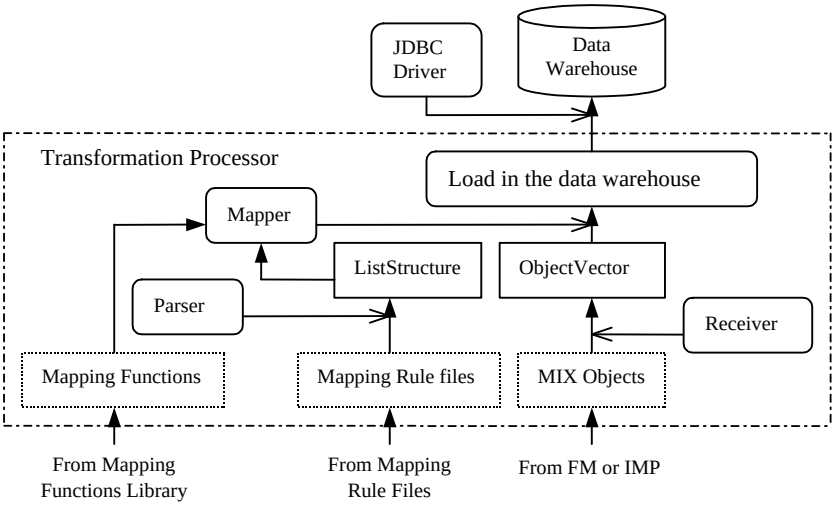

Fig. 7. The Functional Diagram of the Transformation Processor

5 Mapping Rule Language (MRL)

We defined a simple, descriptive language (MRL) to describe the mapping rules, which are specified in mapping rule files. Figure 8 gives an overview of the syntax of MRL. The following examples will explain the usage of MRL in more detail.

DefaultValue: xxx

ClassToDimensionTable (Concept, Tables)-Block
- Generation of the Key
- 1:1-Mapping
  $Ontology.Concept_x$: $ColumnName_x$
- N:1-Mapping
  $F_{agg}(Ontology.Concept_1, ..., Ontology.Concept_n)$: $ColumnName_x$
- 1:N-Mapping
  $F_{dec}(Ontology.Concept_x)$: $ColumnName_1, ColumnName_2, ..., ColumnName_n$
- Call ClassToFactTable-Block (optional)

ClassToFactTable(Concept, Tables)-Block
- The generation of fact-join-key (optional)
- FactLinkDimension-Block : Key columns
- 1:1-Mapping
  $Ontology.Concept_x$: $ColumnName_x$
- N:1-Mapping
  $F_{agg}(Ontology.Concept_1, ..., Ontology.Concept_n)$: $ColumnName_x$
- 1:N-Mapping
  $F_{dec}(Ontology.Concept_x)$: $ColumnName_1, ColumnName_2, ..., ColumnName_n$
- Call ClassToDimensionTable-Block (optional)

FactLinkDimension({Concept,} Fact, Dimensions)-Block
- $DimensionTableKey_x$: $FactTableKey_x$
- Check ($Ontology.Concept_x$)
- FindKeyValue (DimensionTableAttributes): $FactTableKey_x$

Fig. 8. Overview of the Syntax of MRL

1. **MIX concepts to dimension tables**
   Considering the data warehouse in Figure 3 and MIX objects in Figure 5, *BookOffer* is a concept of MIX, while *BookStore* is the name of a dimension table. Mapping rules between MIX objects and dimension tables can be written as follows:

   ClassToDimensionTable(BookOffer, BookStore) {
   Key generated by the system
   Ontology.BookOffer.StoreName : BookStore.Name
   Ontology.BookOffer.URL : BookStore.URL
   Ontology.BookOffer.Availability : BookStore.Availability }

2. **MIX concepts to fact tables**
   Mapping rules between MIX objects and fact tables can be written as follows:

   ClassToFactTable(BookOffer, Discount, Time, Book, BookStore) {
   Discount-fact-join-key generated by system
   DiscountFactLinkTimeDimension : Discount.Time-key
   DiscountFactLinkBookDimension : Discount.Book-key
   DiscountFactLinkBookStoreDimension : Discount.BookStore-key
   Ontology.BookOffer.Price : Discount.Sold-Price
   Ontology.BookOffer.Discount : Discount.Discount }

3. **Connecting dimension tables to fact tables**
Example a: By populating the *BookShop* Fact table, *Time* Dimension, *Customer* Dimension and *Book* Dimension tables using data about books purchase, after a new key value in a dimension table is just generated, we can take it from the dimension tables and assign it to the corresponding foreign key in the fact table.

BookShopFactLinkallDimension(BookShop, Time, Book, Customer) {
Time.Key : BookShop.Time-key
Book.Key : BookShop.Book-key
Customer.Key : BookShop.Customer-key }

Example b: Considering the data warehouse shown in Figure 3, *Time* Dimension based on company's internal data already exists before the *Discount* Fact Table is populated with external Web data. We assume that attributes *Day*, *Month* and *Year* in the *Time* Dimension are continuous, since the books hop receives the order from customers every day. But his competitors announce their new discount book information not at every day, therefore the update date of the discount book information is not continuous. In order to connect the *Time* Dimension to the *Discount* Fact Table, we first find the time key values in those tuples in the *Time* Dimension, whose values of *Day*, *Month*, *Year* equal the *OfferDate* in the *BookOffer* object, then we write these key values in the *Discount* Fact table.

DiscountFactLinkTimeDimension(BookOffer, Discount, Time) {
Check(BookOffer.OfferDate) {
GetDayfromDate(Ontology.BookOffer.OfferDate) : Time.Day
GetMonthfromDate(Ontology.BookOffer.OfferDate) : Time.Month
GetYearfromDate(Ontology.BookOffer.OfferDate) : Time.Year
FindKeyValue(Time.Day, Time.Month, Time.Year) : Time.key
}
Time.key : Discount.Time-key }

The following complete example shows the mapping of a semantic object from Figure 5 to tables in Figure 3.

ClassToDimensionTable(BookOffer, BookStore) {
DefaultValue : "Unknown"
Key generated by the system
Ontology.BookOffer.StoreName : BookStore.Name
Ontology.BookOffer.URL : BookStore.URL
ClassToFactTable(BookOffer, Discount, Time, Book, BookStore) {
Discount-fact-join-key generated by system
DiscountFactLinkTimeDimension(BookOffer, Discount, Time) {
Check(BookOffer.OfferDate) {
GetDayfromDate(Ontology.BookOffer.OfferDate) : Time.Day
GetMonthfromDate(Ontology.BookOffer.OfferDate) : Time.Month
GetYearfromDate(Ontology.BookOfferr.OfferDate) : Time.Year
FindKeyValue(Time.Day, Time.Month, Time.Year) : Time.key }

Time.key: Discount.Time-key }
Ontology.BookOffer.Price:Discount.Sold-Price
DiscountFactLinkBookDimension(BookOffer, Discount, Book) {
Check(BookOffer.Book.ISBN) {
FindKeyValue(Book.ISBN) : Book.Key }
Book.Key : Discount.Book-key }
BookStore.Key : Discount.BookStore-key
Ontology.BookOffer.Discount : Discount.Discount }
If(Ontology.BookOffer.Availability <> NULL)
Ontology.BookOffer.Availability : BookStore.Availability
Else DefaultValue : BookStore.Availability }

6 Related Work

Our system is designed to extract, integrate and transfer Web data into a data warehouse and to interoperate with conventional warehouse data for carrying out meaningful OLAP and supporting the decision making process.

The WHOWEDA system [6, 24] is a Web warehouse that materializes and manages useful information from the Web. Web objects, Web schema and a Web algebra are described by a so-called *Web Information Coupling Model* (WICM). They define a Node type and a Link type to refer to the coupled Web information (HTML or plain text documents, Hyperlinks) and materialize the Web objects in a set of connected, directed graphs. Their focal points are to describe the topological structure of the Web and to design a Web algebra. Our approach differs from WHOWEDA in that a metadata-based object model is used to describe Web information content rather than Web structure. This model is used as the basis to prepare and integrate the data into the Star Schema of a relational data warehouse. Some advantages are that relational data warehouses are already available in many companies and that the same set of tools can be used for data access.

Related with transferring Web data to a star schema of a data warehouse, our research can be compared with DataFoundry [14, 15] and the DWQ project [11, 12].

The DataFoundry project is based on a mediated data warehouse architecture with an ontology infrastructure, it makes extensive use of this infrastructure to generate mediators automatically. However, there are some important differences between our work and theirs. At first, they integrate several existing community databases in a data warehouse, in contrast, we focus on Web data. Comparing with data in scientific databases, Web data in most cases is unstructured or semistructured and has no explicit schema. In our system, MIX provides a mediated model for preparing Web data for mapping them to an existing star schema. An object/relational mapping approach is used to transform data in the mediated model to the star schema of the data warehouse.

In the DWQ project, information integration in the data warehousing environment is studied. They consider two important aspects concerning the design and maintenance of particular data warehouse applications, those are conceptual modeling of the domain, and reasoning support over the conceptual representation. Their approach provides a system-independent specification of the

relationships between sources and between sources and the enterprise model at the conceptual level. But DWQ focuses only on information sources that posses an explicitly specified data schema. Their logical data model is the relational model. In our work, we not only construct domain knowledge representation at the conceptual level, but use self-describing object model at the logical level as well. This offers the possibility to integrate Web data in a semantic correct way. Besides, in [11, 12] the mapping between logical and physical level is not explored. In our system, a file of mapping rules is constructed. With the aid of such a rule file and an object/relational mapping approach we can process transformations between MIX objects and tables of a data warehouse semi-automatically.

7 Conclusions

In this paper, we introduced a Web warehousing approach for integrating a company's own data and Web data in an existing data warehouse. We have shown the benefit of integrating local data with external data, e.g., a competitor's pricing scheme. We discussed our framework for managing and integrating Web data. In this framework we use a self-describing object model that represents data together with metadata that makes implicit assumptions about the structure and semantics of the data explicit. This model provides the basis for integrating Web-based data.

The prepared data can then be transferred to the star schema of a given data warehouse. For this, we have developed a Transformation Processor, which maps MIX objects to dimension and fact tables using mapping rules specified in a mapping file. The system has been implemented in Java.

Current work concentrates on performance improvement. Future work will extend the functionality of the Transformation Processor and the Incremental Maintenance Processor and will address mappings to the multidimensional model.

References

1. Ambite, J. L.; Ashish, N.; Barish, G.; et al.: ARIADNE: A System for Constructing Mediators for Internet Sources, Proc. of the ACM SIGMOD International Conference on Management of Data, Seattle, USA, 1998
2. Anderson, C. R.; Levy, A. Y.; Weld, D. S.: Declarative Web-site Management with Tiramisu, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999
3. Ashish, N.; Knoblock, C. A.; Shahabi, C.: Selectively Materializing Data in Mediators by Analyzing User Queries, Proc. of the International Conference on Cooperative Information Systems, Edinburgh, Scotland, 1999
4. Beeri, C.; Elber, G.; Milo, T.; et al.: WebSuite – A Tool Suite For Harnessing Web Data, Proc. of the International Workshop on the Web and Databases, Valencia, Spain, 1998
5. Bernstein, P. A.; Pal, S.; Shutt D.: Context-Based Prefetch for Implementing Objects on Relations, Proc. of the International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999
6. Bhowmick, S. S.; Madria, S. K.; Ng, W.-K.; Lim, E. P.: Web Warehousing: Design and Issues, Proc. of the International Workshop on Data Warehousing and Data Mining, Singapore, 1998

7. Bornhövd, C.: MIX – A Representation Model for the Integration of Web-based Data, Technical Report, DVS98-1, Department of Computer Science, Darmstadt University of Technology, Nov., 1998
8. Bornhövd, C.: Semantic Metadata for the Integration of Web-based Data for Electronic Commerce, Proc. of the International Workshop on Advance Issues of E-Commerce and Web-based Information System, Santa Clara, USA, 1999
9. Bornhövd, C.; Buchmann, A. P.: A Prototype for Metadata-based Integration of Internet Sources, Proc. of the International Conference on Advanced Information Systems Engineering, Heidelberg, Germany, 1999
10. Calvanese, D.; Giacomo, G. De; Lenzerini, M.; Vardi, M. Y.: Query Answering Using Views for Data Integration over the Web, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999
11. Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; et al.: Description Logic Framework for Information Integration, Proc. of the International Conference on Principles of Knowledge Representation and Reasoning, Trento, Italy, 1998
12. Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; et al.: Information Integration: Conceptual Modeling and Reasoning Support, Proc. of the International Conference on Cooperative Information Systems, New York, 1998
13. Carey, M.; Doole, D.; Mattos, N.: O-O, What Have They Done to DB2?, Proc. of the International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999
14. Critchlow, T.; Ganesh, M.; Musick, R.: Automatic Generation of Warehouse Mediators Using an Ontology Engine, Proc. of the International Workshop on Knowledge Representation meets Databases, Seattle, WA, 1998
15. Critchlow, T.; Ganesh, M.; Musick, R.: Meta-Data based Mediator Generation, Proc. of the International Conference on Cooperative Information Systems, New York, 1998
16. Davulcu, H.; Freire, J.; Kifer, M.; Ramakrishnan, I. V.: A Layered Architecture for Querying Dynamic Web Content, Proc. of the ACM SIGMOD International Conference on Management of Data, Philadelphia, USA, 1999
17. Hackathorn, R. D.: Web Farming for the Data Warehouse, Morgan Kaufmann, 1999
18. Keller, A. M.: Persistence Software: Bridging Object-Oriented Programming and Relational Databases, Proc. of the ACM SIGMOD International Conference on Management of Data, Washington, D.C., 1993
19. Keller, W.: Mapping Objects to Tables, Proc. of European Conference on Pattern Languages of Programming and Computing, Kloster Irsee, Germany, 1997
20. Keller, W.: Object/Relational Access Layers, Proc. of European Conference on Pattern Languages of Programming and Computing, Bad Irsee, Germany, 1998
21. Labrinidis, A.; Roussopoulos, N.: On the Materialization of WebViews, Proc. of the International Workshop on the Web and Databases, Philadelphia, USA, 1999
22. Mattison, R.: Web Warehousing and Knowledge Management, McGraw-Hill, 1999
23. McHugh, J.; Widom J.: Integrating Dynamically-Fetched External Information into a DBMS for Semistructured Data, SIGMOD Record, 26(4), 1997
24. Ng, W.-K.; Lim, E.-P.; Huang, C.-T.; et al.: Web Warehousing: An Algebra for Web Information, Proc. of the IEEE Forum on Research and Technology Advances in Digital Libraries, Santa Barbara, USA, 1998
25. Wiederhold G.: Mediators in the Architecture of Future Information Systems, IEEE Computer, 25(3), 1992
26. Zhu, Y.: A Framework for Warehousing Web Contents, Proc. of the International Computer Science Conference on Internet Applications, Hong Kong, 1999