Managing Source Schema Evolution in Web Warehouses

Adriana Marotta, Regina Motz, Raúl Ruggia
Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay
e-mail: [adriana,rmotz,ruggia]@fing.edu.uy

Abstract

Web Data Warehouses have been introduced to enable the analysis of integrated Web data. One of the main challenges in these systems is to deal with the volatile and dynamic nature of Web sources. In this work we address the effects of adding/removing/changing Web sources and data items to the Data Warehouse (DW) schema. By managing source evolution we mean the automatic propagation of these changes to the DW. The proposed approach is based on a wrapper/mediator architecture, which minimizes the impact of Web source changes on the DW schema. This paper presents this architecture and analyses some novel evolution cases in the context of Web DW.

1. Introduction

The World Wide Web (WWW) has become a major source of information about all areas of interest. Information brokers and global information management systems allow users to classify documents and offer capabilities for retrieving whole documents [1, 2]. However, in order to support high-level decision-making users need to comprehensively analyze and explore the data from such documents. In other words, creating a *Web Warehouse.* Web Warehouses contain consolidated data from many web sources [3, 4].

There are many challenges in creating and maintaining a Web Warehouse. Since web sources range from unstructured to semi-structured text documents and data available in the web has heterogeneous nature the first problem concerns the extraction of data and its translation to a common model. The fact that web sources are developed independently and autonomously rises on a number of differences that must be reconciled when data is brought into the warehouse. The major differences are semantic mismatches across the web sources, such as different data values, different names for the same concepts and differences in how information is represented, i.e. use of tables, items or text. After the warehouse schema is designed, the warehouse must be populated, and over time, it must be kept consistent with web sources. This is a crucial point in web warehouses due to the dynamic nature of web sources and the volatile nature of web data. Furthermore, traditional mechanisms do not solve several problems associated to the nature of web data such as lack of credibility of data, lack of productivity on data analysis, and complex transformation of data to information [5].

The aim of this work is to provide assistance to accomplish those tasks, and more specifically to address the problems of managing evolution of Web Warehouses. In this sense, we present a Web Warehouse architecture that covers the steps starting with information extraction from Web documents and finishing with the DW.

The Web data extraction mechanism gathers information about the user-specified domain from HTML documents and generates mappings from web sources to an integrated schema. We use existing techniques developed in the database area, such as those proposed in [6, 7] for extracting information from web documents [8, 9], and those for integration of database schemas adapted for typical web data conflicts [10]. Finally, a specialized component performs the mapping from the integrated source schema to the web warehouse schema [11], based on existing DW design techniques [12, 13].

The paper focus on the mechanisms for automatically propagating changes occurring on the

structure of the web documents as well as the addition/deletion of web sources. The highly autonomous and volatile nature of Web sources leads to schema evolution in Web Warehouses architectures. Thus, Web Warehouse systems should include mechanisms to manage schema evolution within the integrated schema in order to minimize effects of source changes on web warehouses. We show that this mechanism minimizes disruptions by ensuring that many categories of sources modifications are made transparent to web warehouses users and applications.
The main contribution of this paper is the proposition of a framework for performing semi-automatic propagation of changes on web sources. In this context, we analyse the effects of adding/removing/changing web sources and data items to the warehouse schema.
The rest of the paper is organized as follows. Section 2 presents the general architecture and its components. Section 3 gives a characterization of changes. Section 4 presents details about the managing evolution process through examples. Finally, Section 5 presents some conclusions and future work.

2. System Architecture

Integration is one of the most important aspects of Data Warehouse architectures. We follow the approach to data integration presented in [14] based on a conceptual representation of the data warehouse application domain. The main idea is to declaratively specify suitable matching and reconciliation operations to be used in order to solve possible conflicts among data in different sources.
The typical architecture of these kinds of integration systems is described in terms of two types of modules: *Wrappers* and *Mediators.* The goal of a *Wrapper is* to access a source, extract the relevant data, and present such data in a specified format. The role of a *Mediator* is to merge data produced by different wrappers or mediators.
A distinguishing and relevant issue of Web Warehouses is that sources are mainly external, while sources in classic Data warehouses are mainly internal to the organization. In our Web Warehouse case, the need for maintaining an integrated schema is stronger with respect to other contexts. As a direct consequence, data integration in web warehouses follows the local as view approach, where each data source is defined as a view of a global integrated schema. Moreover, we extend this idea with a notion of differential mediators in order to improve the capabilities of the system in managing source schema changes. Then, the proposed architecture contains two kinds of **Mediators,** which are respectively designed to integrate schemas with and without structural heterogeneities. The former ones are typical mediators that perform schema and instance mappings, the latest ones are mediators that only perform the instance mappings (we called them *Mi).* These *Mi* mediators are useful in case when a new web source is added, in which its data overlaps with an existing one. Figure 1 shows an overall view of the architecture.
The **DW schema** is built starting from the integrated schema. To do this, we apply a framework that transforms relational schemas, and where relational structures are classified according to a Dimensional Model (e.g, a relation can be a *dimension relation* or a *measure relation)* [11].
The **Metadata** contains different kinds of information: (i) The *semantic correspondences* that hold among entities (i.e. classes and relationships) in different schema sources, as well as schema mappings and instance mappings between web sources and the integrated global schema [15]. (ii) The extracted schemas (S_i), which are expressed in terms of ODMG. (iii) The trace that connects structures and changes along the architecture components (from Web sources to the DW). (iv) A classification of the integrated schema structures according to a Dimensional Model [12, 11].

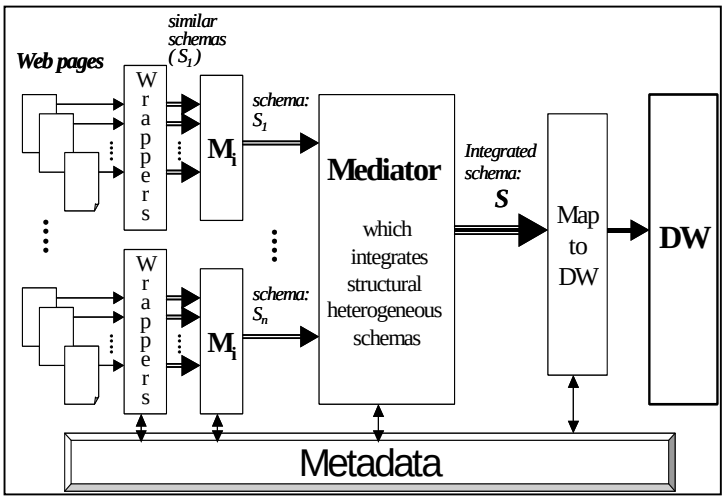

Figure 1. Overall view of the proposed architecture

3. Characterization of Changes

In the context of World Wide Web, an important issue is its *volatility* since data and structure of pages will change at any time. Some sources may disappear while others are added. Thus, **adding/removing Web sources** are crucial changes, which are not thought to occur frequently in traditional Data Warehouse environments.

In addition, and with more generality, we can also identify the following kinds of web document modifications: (a) **Changes that affect display,** i.e. changes on fonts, changes on background colours or changes in the position place of a table in the page. This kind of changes should not be taken as relevant changes since they do not affect data or structure. (b) **Changes that affect web documents structure,** i.e. when tables and lists change their structure by adding or deleting a column, when a table on a page disappears, or new ones are inserted. (c) **Changes that affect the data contents of a document.**

In the context of mediators, the goal is to achieve schema evolution of the already integrated schema when occurs any of the following: (a) **changes in the structure of the local schemas,** or (b) **changes in the semantics of correspondences between the local sources.** The former case is referred to as **structural modifications** while the latter as **semantic modifications.** Moreover, these schema changes occur due to changes on the web level.

From a Data Warehouse point of view, changes may be generated by: (a) **changes on information requirements**, (b) **changes on the data sources** (structural and semantic), and (c) **changes on the refreshment frequency of data sources** [16]. In this work we focus on the second kind of changes: structural and semantic changes on data sources. In the next section we present how to propagate these changes to the DW.

4. Managing Schema Evolution

In this section we analyze the ways to propagate web, structural and semantic changes to the DW. Our goal is the automatic propagation of these kinds of modifications to the DW.

Web to local schemas change propagation

Wrappers are the components that interact with Web sources. In this role, they perform two tasks: (i) extract the data in Web pages, map and structure it to ODMG schemas, and (ii) detect changes in Web pages. These issues have been addressed in [8, 9].

As described in the presentation of the architecture, the two kinds of mediators allow managing in transparent way the addition or changes of sources. This is achieved by adding the new/modified source schema to an already existing mediator *Mi* in spite of the process of schema change propagation to the already integrated schema. Then, we are able to incorporate

this new source in such a way that only the instance mapping need to be actualized.

Local schemas to integrated schema change propagation
Motivated by the fact that some local schema changes may be considered as cases of more elaborated "semantic correspondences" between subschemas [17, 18], our approach is to regard the propagation of local semantic modifications as a form of incremental, semantic schema integration. Our theoretical framework [15] is based on a declarative schema integration methodology, called SIM [19]. A relevant aspect of our work is the (almost) automatic derivation of a new state for the already acquired set of mappings, from the local schemas to the integrated one, due to the occurrence of local semantic modifications.

Integrated schema to DW schema change propagation.
The problem of propagating previous depicted changes to the DW schema includes two main sub-problems: (1) **determining** the changes that must be done to the DW and (2) **applying** the corresponding changes to the DW. In the following, we illustrate relevant cases by means of a motivating example: We consider a DW that provides information about physicians, diseases, drugs, cases of diseases treated by physicians, and medicines. This information is obtained from several different web sites. The extracted information includes name, specialty, and address of the physicians. Different values of address that are extracted from different web sites must be maintained. The DW will enable to analyze the quantity of disease cases treated by the different physicians with different drugs along the time. It is also relevant to analyze the available medicines with their prices according to the different web sites where they are published.

4.1 Determining the changes to the DW
In order to determine the changes that should be applied to the DW schema, we consider the possible changes over the integrated schema and, in some cases, also the web sources' changes that caused them. Besides, we take into account the dependencies existing between DW schema and integrated schema elements.
The Metadata information enables us to distinguish the kinds of changes that occurred along the architecture and to make decisions relative to their propagation to the DW. For this propagation we apply well-known DW design criteria [12], as well as we avoid the generation of NULL values when their existence complicates the multidimensional analysis. We analyse some specific examples that show interesting phenomena in propagating the evolution to DW.

Add relation
This kind of change corresponds to the adding of a new relation to the integrated schema. We present some particular cases, which are shown in Figure 2.
In **(1)** the new relation, `Physician2`, has the same key as relation `Physician` and adds an attribute (`source`), which allows to maintain several values (each one coming from each web source) for the address of one physician. As `address` is a descriptive attribute of the dimension `Physician`, the propagation on the DW schema points to de-normalize, maintaining only one dimension relation `Physician` with all the information.
In **(2)** the new relation in the integrated schema is generated in order to maintain several values for the medicines' prices that come from different sources. The `price` attribute is classified as a measure attribute and the `source` attribute as a dimension attribute. The propagation of this change to the DW generates a new measure relation (`Sale-prices`).
Note that in both cases (1 and 2) the structural schema change was the same, however the change was propagated in different manners because the mechanism takes into account the Dimensional Model semantics of the attributes and relations.

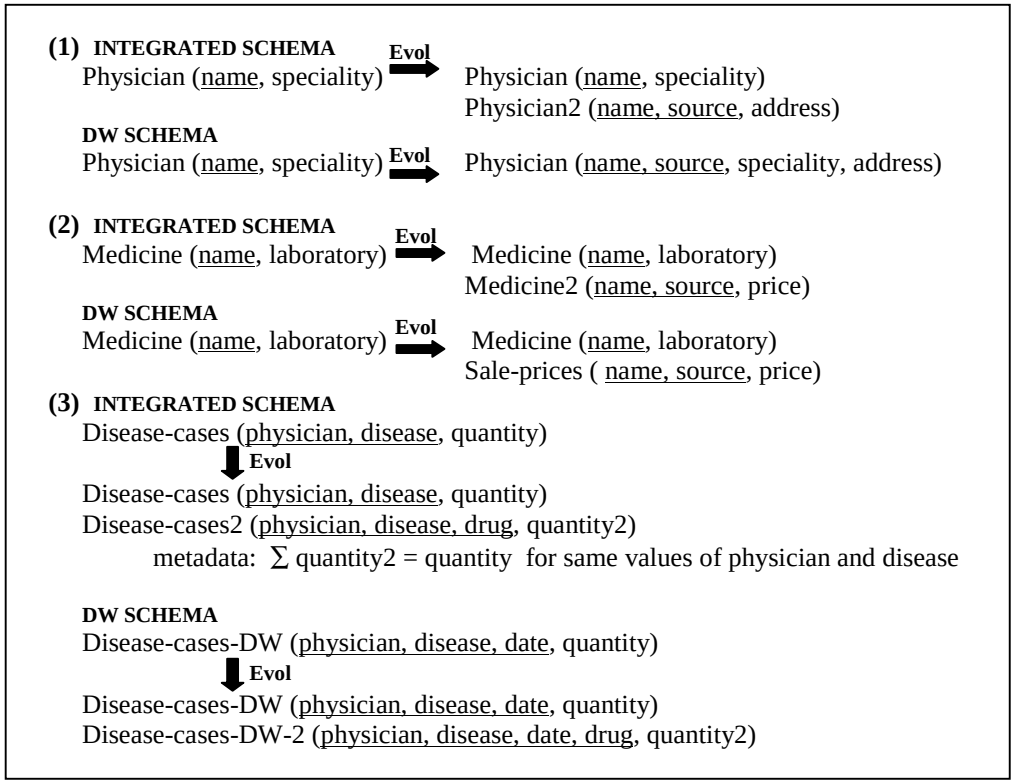

Figure 2. Examples of change Add Relation

In **(3)** the new relation, `Disease-cases2`, is a measure relation that has one more dimension, `drug`, than the already existing measure relation `Disease-cases`. In addition, we know that there is a correspondence of summarization between `quantity` and `quantity2`. As both relations are maintained in the integrated schema, the propagation mechanisms can deduce that not all the sources will provide information related to the dimension `drug`. Therefore, in the DW schema a new measure relation will be generated with the dimension `drug`.

Add attribute

We distinguish some particular cases of attribute adding to the integrated schema. It is important to know whether the attribute was added in all the sources or only in some of them, because null values will be generated if some sources do not provide a value for the added attribute. Consider the following example in Figure 3.

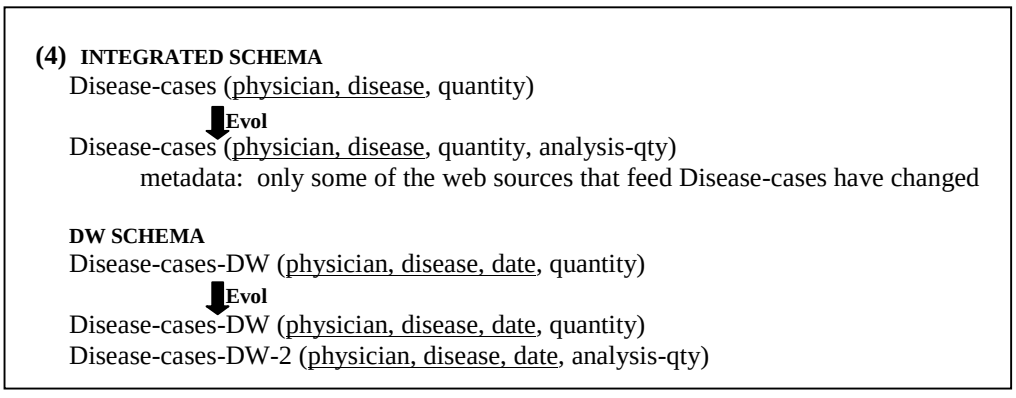

Figure 3. Examples of change Add attribute

In **(4)** a measure attribute `analysis-qty` is added to a measure relation `Disease-cases`, and it is known that the `analysis-qty` will have a NULL value in many of the tuples. The existence of NULL values in a measure attribute would complicate multidimensional analysis. Therefore, the propagation mechanism will generate in the DW a separate measure relation with the new measure attribute (`analysis-qty`).
If an attribute is added to a dimension relation in the integrated schema, the propagation mechanism identifies the derived dimension relation in the DW and adds the same attribute to this relation.

Remove relation and remove attribute
When a relation or attribute is removed from the integrated schema, the propagation mechanism identifies all DW schema elements that were derived from it (depend on it) and determines which action must be applied to them.
The knowledge about dependencies between source and DW schema elements enables to minimize the impact of the changes on the DW schema. For example, consider two attributes, `date` and `month` in the integrated schema; and an attribute `year` in the DW schema, which is calculated using `date`. Suppose that `date` is removed. This change is propagated through modifying the calculation function: it uses the attribute `month` instead of `date`. Then the DW schema is not changed at all.

Remove data item from web source
This is a special case, in which the DW should be affected by a change occurred on the web sources although the integrated schema was not affected by this change. Consider example (5) in Figure 4.

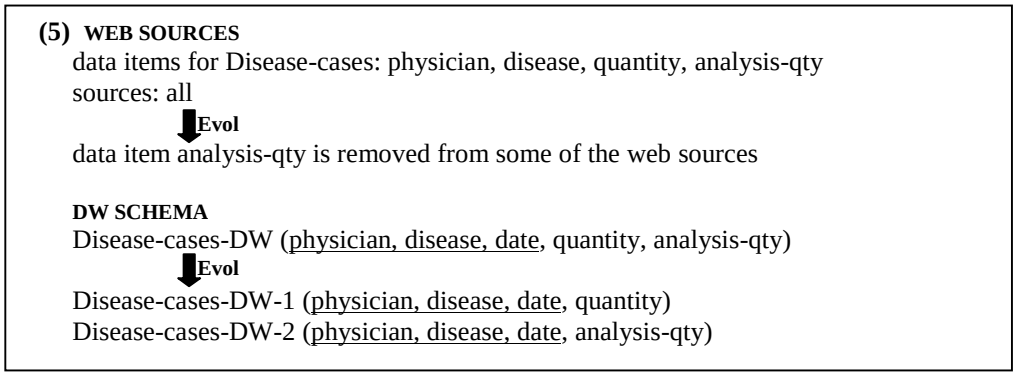

Figure 4. Example of change Remove attribute from web source

In the case of example **(5)** we know that NULL values will start to appear for the measure attribute `analysis-qty`. As commented in example **(4)**, this situation is undesirable for the DW schema. For this reason, this change is propagated to the DW by partitioning the measure relation `Disease-cases-DW` into two measure relations, avoiding the NULL values in measure attributes.

4.2 Applying evolution to the DW
In order to define the evolution strategy we mainly take into account two DW features: (i) a DW stores historical data, (ii) applications that run over the DW only query the data, they do not modify it. These two features lead us to apply a version-based approach to evolution [20, 21]. Concerning the first one, the application of an adaptational approach [22] could cause the lost of historical data due to possible removes of subschemas. Concerning the second feature, in a version-based approach, the only write that is applied to the DW consists of refreshments of data, which are applied uniquely to the last version of the DW. Thus, it will not be

necessary to convert updates from one structure to another.
Therefore, we propose a version-based mechanism, in which queries that were already running over any version can continue running over the same version without any modification.

5. Conclusion

The main contribution of the paper lies in the provision of a framework for semi-automatic propagation of source schema changes and addition/deletion of sources to an already existing DW. More concretely, the proposed architecture with the two kinds of mediators allows managing some addition or changes of sources in a transparent way. In addition, we have described some novel cases of changes such as source changes that do not impact the integrated schemas but affect the DW (Section 4.1, Example 5) and semantic changes that enable to infer the dimensional classification of an attribute (Section 4.1, Example 3). A way to trace evolution propagation is through the metadata. It plays an important role in managing evolution. More specifically, as seen in the examples, classifying the attributes according to a Dimensional Model enables a more effective treatment of evolution.
In the CSI Group[1] we are currently developing a prototype of environment for DW design and evolution management. In this context we are developing rules that enable an automatic classification of evolved attributes according to a Dimensional Model. These rules take into account the kind of changes that occur in the correspondence assertions associated to the attributes.

References

[1] O. Etzioni. *The World-Wide Web: Quagmire or Gold Mine?* CACM, 39(11):65-68, November 1996.
[2] Susan Malaika. *Resistance is Futile: The Web Will Assimilate Your Database.* Data Engineering Bulletin 21(2): 4-13, (1998)
[3] Richard D. Hackathom. *Web Farming for the Data Warehouse.* The Morgan Kauimann Series in Data Management Systems, Jim Gray, Series Editor. November 1998. ISBN 1-55860-503-7
[4] L. Faulstich, M. Spiliopoulou, V. Linnemann. *WIND: A Warehouse for Internet Data.* Proceedings of British National Conference on Databases , pp. 169-183, London, 1997.
[5] Sourav S. Bhowmick. *Whom: A Data Model and Algebra for a Web Warehouse.* Phd Thesis, Nanyang Technological University, Singapore. August 2000.
[6] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, A. Crespo. *Extracting Semistructured Information from the Web*. Proc. of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.
[7] G. Huck, P. Fankhauser, K. Aberer, E. Neuhold. *JEDI: Extracting and Synthesizing Information from the Web*. COOPIS 98, New York, August, 1998. IEEE Computer Society Press.
[8] A. Gutiérrez, R. Motz, D. Viera. *Building a database with information extracted from web documents.* In Proc. of the International Simposio de la Sociedad Chilena de Computación, SCCC 2000, Santiago , Chile, November 2000.
[9] J. Ferreiro, R. Motz, F. Perelló, D. Wonsever. *Generación Automática de una Base de Datos desde Documentos de la Web.* Congreso Argentino de Ciencias de la Computación, CACIC 2000, Ushuahia, October 2000.
[10] A. Do Carmo. *Aplicando Integración de Esquemas en un contexto DW-Web.* Master thesis. Universidad de la República. Uruguay. 2000.
[11] A. Marotta. *Data Warehouse Design and Maintenance through schema transformations.*

[1] CSI Group of the Instituto de Computación - Universidad de la República, Uruguay.

Master thesis. Universidad de la República. Uruguay. 2000.
[12] R. Kimball. *The Data Warehouse Toolkit.* J. Wiley & Sons, Inc. 1996
[13] L. Silverston, W. H. Inmon, K. Graziano. *The Data Model Resource Book.* J. Wiley & Sons., 1997
[14] Calvanese D., De Giacomo G., Lenzerini M., Nardi D. and Rosati R. *A principled Approach to Data Integration and Reconciliation to Data Warehousing.* Proc. of the int. Workshop on design and Management of Data Warehouses (DMDW99) Heidelberg, Germany, June 1999.
[15] R.Motz. *Dynamic Mantainance of an Integrated Schema,* PhD Thesis, Darmstadt University of Technology, Germany (forthcomming).
[16] Quix C. *Repository Support for Data Warehouse Evolution.* Proc. of the int. Workshop on design and Management of Data Warehouses (DMDW99) Heidelberg, Germany, June 1999.
[17] Regina Motz. *Propagation of Structural Modifications to an Integrated Schema.* Proceddings of Advanced Database Information Systems, ADBIS'98, Poland, September 1998.
[18] Regina Motz and Peter Fankhauser. *Propagation of Semantic Modifications to an Integrated Schema.* Proceedings of Cooperative Information Systems, Coopls'98, New York, August 1998.
[19] Regina Motz, Peter Fankauser and Gerald Huck. *Schema Integration.* Deliverable IRO-DB(P8629) D4-4/1 - 1995.
[20] F. Ferradina, S. Lautemann. *An Integrated Approach to Schema Evolution for Object Databases.* OOIS 1996, London, U.K.
[21] S. Lautemann. *An Introduction to Schema Versioning in OODBMS.* In Proc. of the 7th DEXA Conference, Zurich Switzerland, September 1996. IEEE Computer Society. Workshop Proceedings.
[22] F. Ferradina, R. Zicari. *Object Database Schema Evolution: are Lazy Updates always equivalent to Immediate Updates?* OOPSLA Workshop, September 1993, Washington D.C.