Managing Source Schema Evolution in Web Warehouses

Adriana Marotta

Instituto de Computación - Universidad de la República
Fax: 7110469 - C.P.: 11300
adriana@fing.edu.uy

Regina Motz

Instituto de Computación - Universidad de la República
rmotz@fing.edu.uy

Raul Ruggia

Instituto de Computación - Universidad de la República
ruggia@fing.edy.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 reduces the impact of Web source changes on the DW schema. This paper presents this architecture and analyses some selected evolution cases in the context of Web DW.

Keywords: *Web Warehouses, Data Warehouses, Schema Evolution, Interoperability*

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. This motivate the creation of *Web Warehouses*, which are Data Warehouses (DW) that contain consolidated data obtained from web sources [3, 4].

There are many challenges in creating and maintaining Web Warehouses. 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 critical 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 tasks that are performed in the design phase as well as the ones that are performed in the maintenance of the Web Warehouse need the support of the *metadata*. Metadata is the information about the data and the processes that are applied to it. We need, for instance, metadata about the classification of web pages, the mappings between the different representations of the information, the correspondences between concepts (used for integration), and the DW design strategies.

This paper focus on the mechanisms to cope with changes occurring on the structure of the web documents as well as the addition/deletion of web sources. These changes lead to schema evolution in Web Warehouse architectures. Due to the highly evolutive nature of Web sources, Web Warehouse systems should include mechanisms to manage schema evolution in order to minimize effects of source changes on Web Warehouses. We propose a mechanism that reduces 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 to the DW schema. In this context, we analyze 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 related work. Section 3 presents the general architecture and its components. Section 4 gives a characterization of changes. Section 5 concentrates on the propagation of source changes along the architecture. Section 6 presents evolution management in the Web Warehouse, through examples. Finally, Section 7 presents the conclusions.

2 Related Work

Few proposals of Web Warehouses focus on schema evolution problems. This lead us to analyze works about change management in Web systems, Schema Evolution, as well as Web Warehouses in general.

The work in [14] consist of a Web Warehousing system, which stores information about the graphs representing web pages and links among them. This work do not focus on interpreting and extracting the content of each page. This kind of approach is also followed in [15, 16] where they present a tool to help users to navigate on Web sites making use of the RDF [17] metadata approach. This tool provides a mechanism to represent a Web site as a graph, whose nodes can represent any kind of hierarchy or associated links.

The work in [18, 19] present a framework for warehousing web data, where *wrappers* extract information and map it to MIX (Metadata based Integration Model for data Xchange). A *federation manager* integrates the heterogeneous data interacting with an *ontology server*. Finally, a *transformation processor* maps MIX objects to an existing relational *star schema* (to the DW). These works focus on the transformations from MIX objects to warehouse tables. They do not address the problem of web sources dynamicity and the propagation of the changes to the web warehouse.

In [20], there is an analysis about the involved problems on the evolution management of web-based information sources. They study the possible source changes and how the wrappers, view definitions and extent of the Data Warehouse could be adapted. In [21] the authors study the problem of view adaptation and synchronization, presenting a taxonomy of the existing problems. These two works refer to the EVE system [22] and comment how these issues are solved in this context.

There exist tools for solving the problem of detection of changes in the Web. Some of them only notify when a change occurs [23], others deduce the occurred changes from two versions of a web page [24, 25], and finally there exist others that notify that a change occurred and offer information about the change [26, 27, 28, 29].

For describing descriptive and semantic metadata about the web in a standard way, the W3C (WWW Consortium) [30] proposes RDF [17] and RDFS [31], which provide mechanisms for automatically sharing Web knowledge. In [15], RDF is used for representing Web sites. In [32, 33] they work with semantic metadata about the construction of the DW. However it is important to catalogue all this information in a standard way, so that it can be inter-changed with other systems. CWM [34] is a standard for metadata that may be used in this context.

3 System Architecture

Our architecture 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.

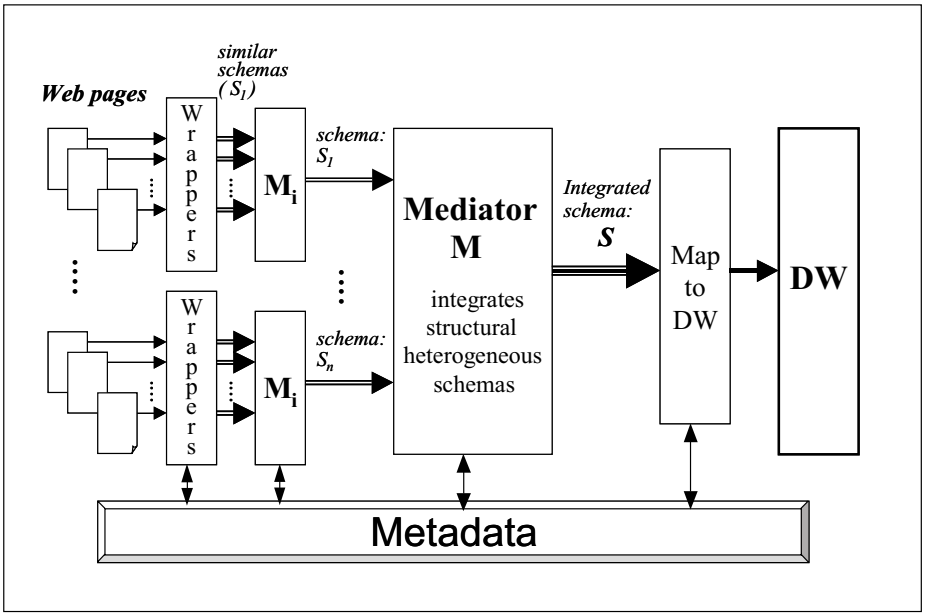

Figure 1: *Overall view of the proposed architecture*

Figure 1 shows an overall view of the architecture. The set of **Web pages** is grouped according to the information we extract from them. For each group of pages we define one schema structure and an associated **wrapper**. The wrappers extract the data in the pages, then they map and structure it to ODMG schemas. The output of each of them is a set of identical schemas populated with the corresponding information from each page. The wrappers also detect changes in Web pages. These issues have been addressed in [8, 9].

A distinguishing and relevant issue of Web Warehouses is that sources are mainly external, while sources in classic DWs are mainly internal to the organization. Then, it is very important to maintain an integrated schema, which represents a unified view of the data. As a direct consequence, data integration 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 by introducing differential mediators in order to improve the capabilities of the system to manage source 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 (we called them *M)*, while the latter ones are mediators that only perform the instance mappings (we called them *Mi).* Each *Mi* mediator integrates the results returned by each wrapper and each *M* mediator integrates the schemas and data returned by the *Mi* mediators.

In order to solve data conflicts in information integration we follow the approach presented in [35], which is 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 **DW schema** is designed by applying a set of high-level schema transformations to the integrated schema. These schema transformations embed meaningful DW design techniques [12, 13]. The whole DW design framework is presented in [11]. Then, the designer applies these transformations according to the desired design criteria, such as "de-normalize dimensions". According to this transformation based framework, database tables are classified according to a Dimensional Model (e.g, a relation can be a *dimension relation* or a *measure relation)*. The transformation based approach facilitates the construction of structures that are specifically designed to satisfy DW's requirements, and also provides design traceability.

Traceability is a quality property that enables to improve design process as well as DW management. Concerning the design process, the trace behaves as a valuable documentation of the design and it can be very useful for design process reuse. Concerning DW management, the trace is an valuable tool for obtaining the mapping between source and DW schema elements. This mapping is necessary at least for solving the following three problems in DW

management: (i) data loading processes, (ii) source schema evolution, and (iii) error detecting.

Finally, the **Metadata** contains different kinds of information: (i) Web pages classification criteria, (ii) mappings between the web pages and integrated global schema [38], and between the integrated schema and the DW [11], (iii) semantic correspondences that hold among entities in the different source schemas (used for integration) [39], (iv) the classification of the DW schema structures according to the Dimensional Model [11, 12], and (v) DW design criteria [11].

4 Characterization of Changes

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

We intend to characterize the changes that may occur at the different layers of our architecture.

In general, **Web changes** can be classified into two main groups: (1) *changes in the set of source pages* and (2) *changes in the content of the pages*. In the first group we consider the following changes: (a) *add page*, and (b) *remove page*. In the second, we distinguish the following kinds of changes: (a) *changes that affect display*, (b) *structural changes*, (c) *changes in the data contents of the page*, and (d) *semantic changes*. Display changes (2a) refers, for example, to changes on fonts, changes on background colors 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. Structural changes (2b), refer to the representation of concepts in the page, e.g. tables and lists change their structure by adding or deleting a column, a table on a page disappears, or new ones are inserted. Data content changes (2c) refers to the changes on the data provided in the page, without changing the structure of the page. A change of this kind would be, for instance, adding a new doctor to a list of doctors in a page. (In this work we do not focus on this kind of changes.) Semantic changes (2d), are those where the page keeps the same structure, but the data contained in it does not correspond to the same real-world concept as before. For example, a page refers to medical doctors in general, and it changes, starting to refer only to medical doctors who are cardiology specialists.

At the **wrappers** level, we distinguish three kinds of changes: (1) *remove wrapper*, (2) *change on the mappings*, and (3) *semantic change*. When a group of pages contains only one page and this page is removed, then the corresponding wrapper is also removed (case 1). Eventually, we could disconnect the wrapper instead of removing it, since it could be useful in the future. Schema and instance mappings between Web pages and the wrappers may also change (case 2) as a consequence of changes on Web pages. A semantic change on a wrapper (case 3) is a change on the semantics of a schema element (class or attribute) that belongs to a schema that is generated by the wrapper.

In the context of **mediators**, we distinguish changes that can occur over *Mi* mediators from changes over *M* mediators. Changes on *Mi* mediators can be classified in: (1) *remove mediator*, (2) *update instance mappings*, and (3) *semantic change*. Cases (1) and (3) are analogous to the ones of wrappers. Case (2) refers to the instance mapping existing between the wrappers' schemas and the *Mi* mediator. We classify changes on *M* mediators in: (1) *schema change*, (2) *update correspondences*, and (3) *update mappings*. Due to changes occurred on a *Mi*, such as a semantic change or a removal, the integrated schema of *M* may suffer a change (case (1)). Also the semantic correspondences defined between the *Mi* schemas and the instance and schema mappings with the *Mis* may change (case (2) and (3)).

Changes on the **DW** are not only caused by changes on the sources, we distinguish three cases: (a) changes on information requirements, (b) changes on the data sources (changes on the mediator's schema or on web sources structure/data contents), and (c) changes on the refreshment frequency of data sources [37]. In this work we focus on the second, i.e. changes on the data sources. The possible changes over the DW are: (1) *schema change*, and (2) *update mappings*. (1) refers to changes over the relational schema of the DW. (2) refers to changes that have to be applied to the mappings between the *M* mediator's schema and the DW schema.

Table 1 shows the possible changes that may occur at the different layers of the architecture.

	Add	Remove	Display	Structural	Semantic	Data Contents	Schema	Mappings	Correspondences
Web pages	X	X	X	X	X	X			
Wrapper		X			X			X	
Mi		X			X			X	
M							X	X	X
DW							X	X	

Table 1: Changes according to the architecture

In next section we concentrate on the propagation of the changes from the Web to the DW.

5 Propagation of Web Evolution

When there is a change over the Web sources the Web Warehouse has to be updated in order to maintain the consistency with the sources. In this section we analyze the ways to propagate web changes to the DW. Our goal is the automatic propagation of these kinds of modifications minimizing the impact on the DW.

5.1 Overview of the Process of Change Propagation

In the process of change propagation from the sources to the DW the different components of the system are affected. The wrappers and mediators should "absorb" the changes as much as possible so that the impact on the DW is minimized. Globally, we can distinguish in the propagation process 3 stages that are solved through different mechanisms:

1) Web to export schemas change propagation

As described in the presentation of the architecture, the distribution of Web pages in groups and the two kinds of mediators facilitate managing in transparent way the addition or changes of sources. This is achieved by adding the new/modified page to an existing wrapper and the new schema to an already existing mediator *Mi*, in spite of following the process of schema change propagation to the already integrated schema. Then, we are able to incorporate new sources in such a way that only the mappings need to be updated.

2) Export 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 sub-schemas [38, 39], our approach is to regard the propagation of local semantic modifications as a form of incremental, semantic schema integration. Our theoretical framework is based on a declarative schema integration methodology, called SIM [36]. 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. This approach confronts with the hypothetical use of traditional integration methodologies, i.e. non-incremental ones, to propagate local schema evolution. Traditional integration methodologies always require as input the whole set of correspondences between the local schemas and a complete reintegration every time a semantic modification occurs. The utilization of an incremental schema integration methodology, by contrast permits to provide new correspondences in an incremental manner and re-integrate only the subschemas that are affected by these new correspondences.

3) 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 applied to the DW* and (2) *applying the changes to the DW*.

In order to determine the changes that should be applied to the DW schema in each case, we consider the possible changes over the integrated schema and, in some cases, also the web sources changes. Given an attribute or relation of the integrated schema, the *design trace* allows us to identify all DW schema attributes and relations that were derived from it. In [11] the DW source evolution problem is addressed through deducing all dependencies between integrated schema elements and DW schema elements. Furthermore, [11] introduces a set of Propagation Rules that establish the actions that must be performed in each case of

change to the integrated schema and dependency between integrated-schema and DW elements. We also take into account the classification of the schema attributes and relations according to multidimensional concepts.

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. Later in the document, we analyse some specific examples that show interesting phenomena in propagating the evolution to DW.

We also define a strategy for applying the changes to the DW (presented in Section 6.2).

In Table 2 we show the possible effects of each web change on: the wrapper, the *Mi* mediator, the *M* mediator and the DW. The components of these layers that may be affected are: schema, schema mappings, instance mappings, semantics, and correspondences. The operations that may be applied to these components are: add, remove, and update or change.

We distinguish semantic changes that occurred in *one page* from semantic changes that occurred in *all pages of a group*, because these two cases are managed in different ways. The former is the case when a page is changed to another wrapper, and the latter generates other kind of changes.

For simplicity we use the following abbreviations:
S – Schema
M – Mapping
I – Instance
C – Correspondence
Sem - Semantic

Web Change	Wrapper	Mi Mediator	M Mediator	DW
Add Web page.	Add SM	Update IM	No Change	No Change
Remove Web page.	No Change	Remove IM	No Change	No Change
	Remove Wrapper	Remove Mediator	S Change + Update SM	No Change
				S Change + Update SM
			Remove C	No Change
			Remove C + S Change + Update SM	No Change
				S Change + Update SM
Semantic change in a group of web pages.	Sem Change	Sem Change	C Change + S Change + Update SM	No Change
				S Change + Update SM
Semantic change in a web pages.	*	*	*	*
Structural change in a web pages.	Update SM	No Change	No Change	No Change
				S Change + Update SM
	*	*	*	*

Table 2: Propagation of Web changes from web sources to DW

Note: The rows filled with "*" correspond to the cases where the Web page is changed to another existing Wrapper. This change is equivalent to the changes "Remove Web page" and "Add Web page" consecutively applied.

When a web page is added to a group of pages the wrapper and the first-level mediator are updated. A schema mapping is added to the wrapper and the mediator's instance mapping is updated.

When a web page is removed we can have two different situations: (i) the page belonged to a group of pages that has other pages or (ii) the page was the only page of a group, so the group is empty now. In Table 2 it can be seen that there are two paths from the change "Remove web page". In one of them the wrapper and its corresponding mediator are removed (situation ii). In this case, a better solution could be to "disconnect" them instead of removing them, so that in the future they can be used again. In the M mediator we have three possibilities according to the answers to the following questions: (i) was there any correspondence between the removed schema and the other schemas?, and (ii) must the integrated schema be modified? Finally, the integrated-schema change may cause a change in the DW schema.

A semantic change in a group of web pages causes a change in the correspondences and in the integrated schema at the M mediator. This also may cause a change in the DW schema.

When there is a change in the structure of a web page it may be changed to another group of pages (another wrapper) or it may be kept in the same group. In the latter case there is a schema-mapping update in the corresponding wrapper. At the DW level there may be a schema change, for example when the change in the page causes that certain data is not provided any more.

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.

5.2 An Example

In the following example we show the propagation of a semantic change in a group of Web pages.

Figure 2 presents the information and schemas maintained in each layer of the system. Figure 3 shows the information and schemas in each layer after the change of the Web pages.

WEB:
Information provided in Web pages, group 1:

physician-name, address, speciality, disease, quantity of cases per disease

Information provided in Web pages, group 2:
disease, causes, treatment, medical doctor, quantity of cases

WRAPPER and Mi MEDIATOR:
W1, Sub-Schema 1:
Physicians(name, address, speciality)
Disease-Cases(physician, disease, quantity)

W2, Sub-Schema 2:
Diseases(name, causes, treatment)
Cases(doctor, disease, quantity)

M MEDIATOR: (only integration of Disease-Cases and Cases is shown)

Correspondences:
physician $\equiv$ *doctor*
disease $\equiv$ *disease*
quantity $\equiv$ *quantity*

Integrated schema:
Disease-cases (physician, disease, quantity)

DATA WAREHOUSE

DW schema:
Disease-cases-DW (physician, disease, date, quantity)

Figure 2: Web information maintained in the Warehouse

In this example we have 2 groups of web pages. In the first group, information about physicians and cases of diseases is extracted. In the second, information about diseases and cases treated by physicians is extracted.

Figure 2 shows the schemas of the Mi mediators corresponding to each group. Possible data conflicts between different pages of the same group are solved at this stage. At the M mediator the schemas and information returned by the previous layer are integrated. Note that from this point to the DW the figure only shows the sub-schemas corresponding to the disease cases. The M mediator uses, among others, the equivalence correspondence between *quantity* of *sub-schema 1* and *quantity* of *sub-schema 2*. It contains the relation *Disease-cases* with attributes *physician, disease* and *quantity.* Finally, the corresponding DW relation *Disease-cases-DW* contains the attributes *physician, disease, date,* and *quantity.*

WEB:
Information provided in Web pages, group 1:

physician-name, address, speciality, disease, quantity of cases per disease

Information provided in Web pages, group 2:
disease, causes, treatment, medical doctor, quantity of tests per patient

WRAPPER and Mi MEDIATOR:
W1, Sub-Schema 1:
Physicians(name, address, speciality)
Disease-Cases(physician, disease, quantity)

W2, Sub-Schema 2:
Diseases(name, causes, treatment)
Disease-Tests(doctor, disease, quantity)

M MEDIATOR:

Correspondences:
physician ≡ *doctor*
disease ≡ *disease*
quantity ≠ *quantity*

Integrated schema:
Disease-cases (physician, disease, quantity, tests-quantity)

DATA WAREHOUSE

DW schema:
Disease-cases-DW (physician, disease, date, quantity)
Disease-cases-DW-2 (physician, disease, date, tests-qty)

Figure 3: Propagation of the Web change to the Warehouse

Figure 3 shows that there is a change in the semantic of the information provided by the pages in group 2. The *quantity* information, which previously referred to the quantity of cases of a disease, now refers to the quantity of tests that are made to the patients suffering the disease. This change is automatically assumed by the Mi schema corresponding to group 2, which remains with the same structure. However, at the M mediator there is a change in the schema, since the equivalence correspondence between the attributes *quantity* is not valid any more. Now the relation *Disease-cases* of the integrated schema has two different *quantity* attributes: *quantity* and *tests-quantity*. At the DW level there is also a schema change. In order to maintain the two different *quantity* attributes two relations are created instead of one. This is because the relation and attributes are classified as measure. We explain this reason more in depth in next section.

6 Web Warehouse Evolution

In this section we concentrate on the problem of determining Web Warehouse schema changes generated by source changes, and applying them.

We first analyze some particular cases of changes, describing only the modifications that are applied to the DW, but not *how* the schema and instance are modified. Afterwards, we describe the strategy we use for applying evolution to the DW.

6.1 Analysis of Particular Cases

We present some particular cases of changes of the *M* mediator schema and how they are propagated to the DW. We work with Relational Model for both mediator and DW schemas. We also present a case of a web source change that does not affect the mediator schema, but, however, affects the DW.

Case A: Add relation to M schema

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.

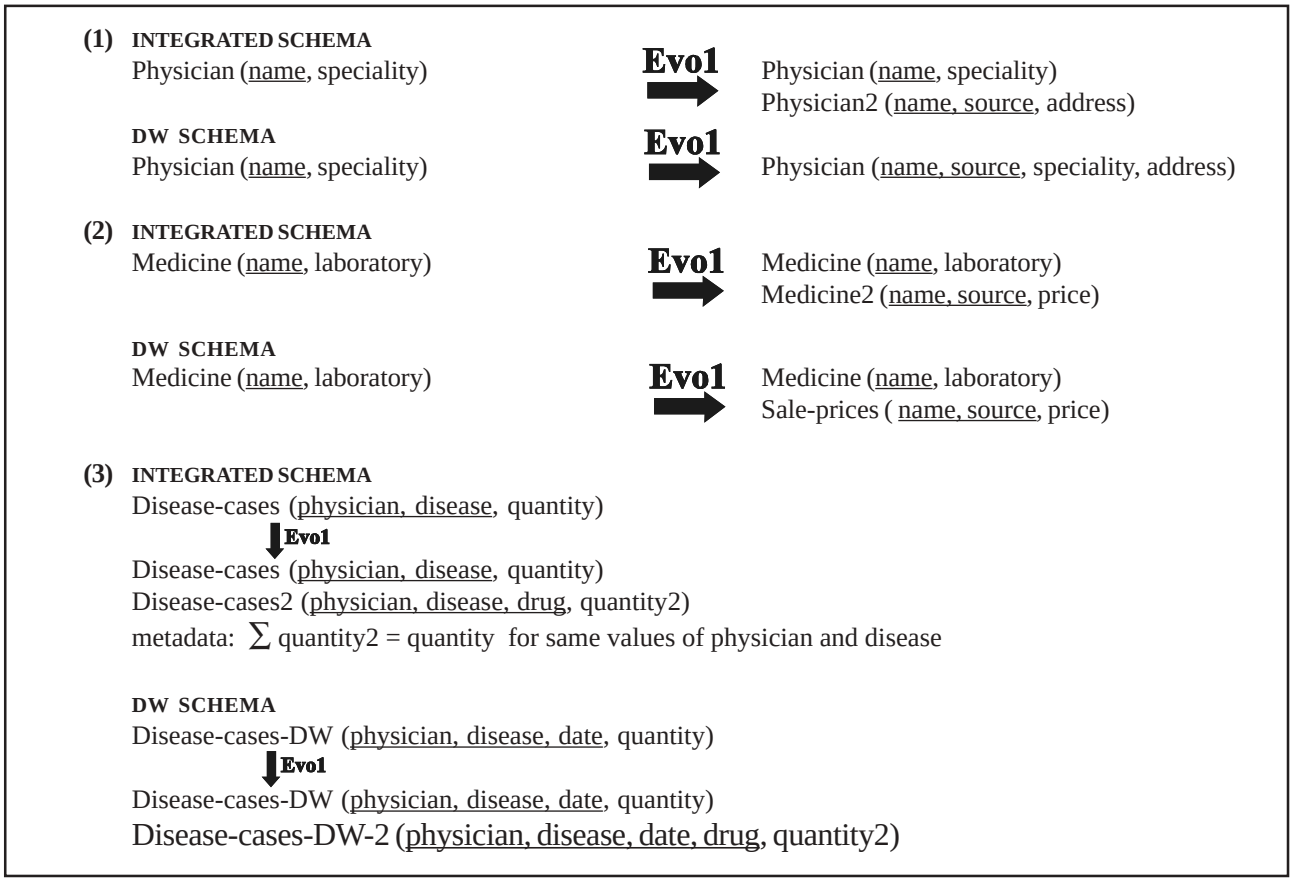

Figure 4: Examples of change *Add relation to M schema*

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.

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.

Case B: Add attribute to M schema

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.

In **(4)** a measure attribute tests-qty is added to a measure relation Disease-cases, and it is known that the tests-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 (tests-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.

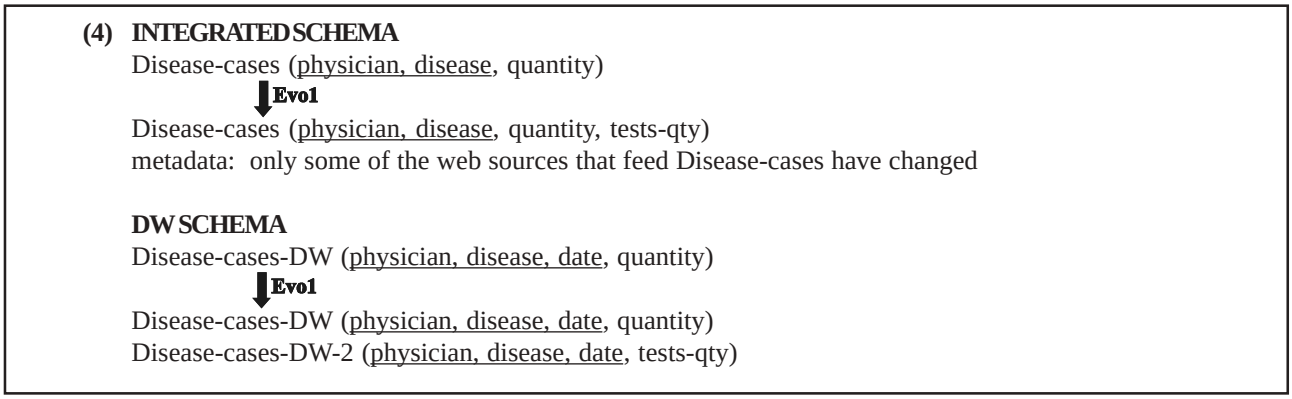

Figure 5: Examples of change *Add attribute to M schema*

Case C: 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.

Case D: 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.

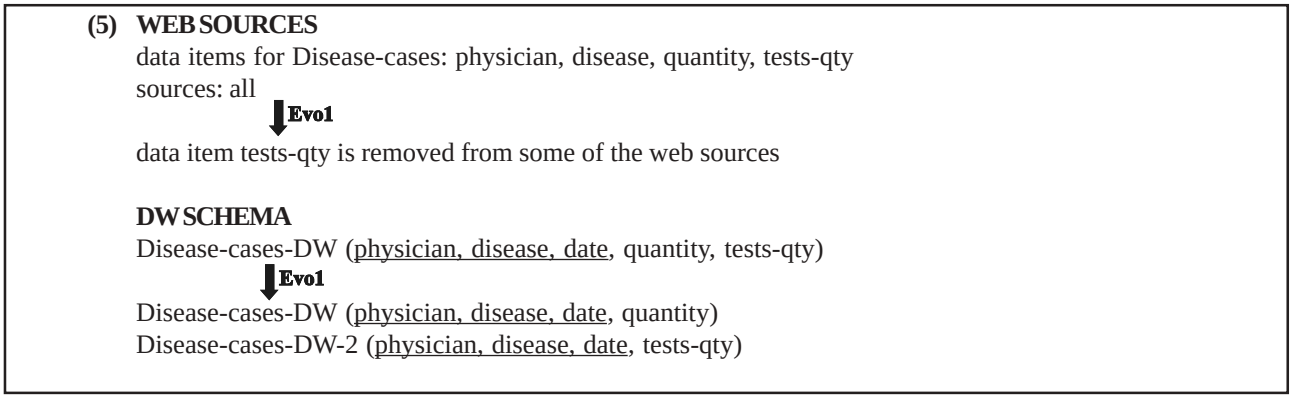

Figure 6: Example of change *Remove data item from Web source*

In the case of example **(5)** we know that NULL values will start to appear for the measure attribute tests-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.

6.2 Applying evolution to the Web Warehouse

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 [40, 41]. Concerning the first one, the application of an adaptational approach [42] 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.

7 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. Mar 2000.

[11] A. Marotta. Data Warehouse Design and Maintenance through schema transformations. Master thesis. Universidad de la República. Uruguay. Oct 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] E.-P. Lim, W.-K. Ng, S. S. Bhowmick, F. Qin and X. Ye. A Data Warehousing System for Web Information. *East Meets West: The First Asia Digital Library Workshop*, The University of Hong Kong, Hong Kong, Aug. 6 - 7, 1998.

[15] A. Moura, C. Fernandes. A Metadata Approach to Represent and Visualize Sites on the Web. *Proc. of the International Workshop on Information Integration on the Web (WIIW).* Rio de Janeiro, Brazil, Apr 2001.

[16] C. S. Fernandes. A Graphic Tool for Representing and Visualizing Sites on the Web. Master Thesis. IME, Rio de Janeiro, Brazil, Feb 2001.

[17] Resource Description Framework. Model and Syntax Specification. W3C Recommendation. Feb. 1999. http://www.w3.org/TR/REC-rdf-syntax/.

[18] Y. Zhu, C. Bornhovd, D. Sautner A. Buchmann. Materializing Web Data for OLAP and DSS. *First International Conference on Web-Age Information Management (WAIM'00).* Shanghai, China, June 21-23, 2000.

[19] Y. Zhu, C. Bornhövd, and A. Buchmann. Data Transformation for Warehousing Web Data. *In the proceedings of 3rd Intl. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems(WECWIS 2001),* June 2001, San Jose, USA.

[20] E. A. Rundensteiner, A. Koeller, X. Zhang. Maintaining Data Warehouses over Changing Information Sources. *Communications of the ACM*. June 2000 / Vol. 43 No. 6

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

[21] E. A. Rundensteiner, A. J. Lee, A. Nica. On Preserving Views in Evolving Environments. *In Proceedings of 4th. Int. Workshop on Knowledge Representation Meets Databases (KRDB'97): Intelligent Access to Heterogeneous Information*. Athens, Greece. Aug. 1997, pp. 13.1-13.11.

[22] A. J. Lee, A. Nica, E. A. Rundensteiner. The EVE framework: View Evolution in an Evolving Environment. Technical Report WPI-CS-TR-97-4, Worcester Polytechnic Institute, Dept. of Computer Science, 1997.

[23] Internet FastFind. PCMagazine. The 1998 Utility Guide – Off-line search utilities. (http://www.zdnet.com/pcmag/features/utility/offsch/uos4.htm)

[24] HTML Compare. Oct 2001. http://www.htmlcompare.com

[25] HTMLDiff. Oct 2001. http://www.htmldiff.com

[26] Mind-It. Oct 2001. http://www.netmind.com

[27] BuzzCity. Oct 2001. http://es.buzzcity.com

[28] L. Liu, C. Pu, W. Tang. WebCQ – Detecting and Delivering Information Changes on the Web. *CIKM* 2000. Mc. Lean VA. USA.

[29] Change Tracking. http://www.wisosoft.com

[30] The World Wide Web Consortium. 2000. http://www.w3c.org.

[31] Resource Description Framework Schemas. W3C Proposed Recommendation. 2001. http://www.w3.org/TR/1998/WD-rdf-schema

[32] A. Moura, M. C. Victorino. Using Mediator and Data WarehouseTechnologies for developing an Environmental Decision Support System. *Proc. of the International Workshop on Information Integration on the Web (WIIW).* Rio de Janeiro, Brazil, Apr 2001.

[33] M. C. Victorino. Use of Mediation Technology to Extract Data and Metadata on the Web for Environmental Decision Support Systems. Master Thesis, IME, Rio de Janeiro, Brazil. Feb 2001.

[34] Common Warehouse Model. 2001. http://www.omg.com

[35] 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.

[36] Regina Motz, Peter Fankauser and Gerald Huck. Schema Integration. Deliverable. *IRO-DB*(P8629) D4-4/1 - 1995.

[37] 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.

[38] Regina Motz. Propagation of Structural Modifications to an Integrated Schema. *Proceddings of Advanced Database Information Systems, ADBIS'98*, Poland, September 1998.

[39] Regina Motz and Peter Fankhauser. Propagation of Semantic Modifications to an Integrated Schema. *Proceedings of Cooperative Information Systems, Coopls'98*, New York, August 1998.

[40] F. Ferradina, S. Lautemann. An Integrated Approach to Schema Evolution for Object Databases. *OOIS* 1996, London, U.K.

[41] 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.

[42] F. Ferradina, R. Zicari. Object Database Schema Evolution: are Lazy Updates always equivalent to Immediate Updates? *OOPSLA Workshop*, September 1993, Washington D.C.