How to Guarantee Analysis Results Coherence after Data Warehouse Schema Changes Propagation towards Data Marts?

Noura Azaiez and Jalel Akaichi
Department of Computer Science, ISG-University of Tunis, Le Bardo, Tunisia

Keywords: Data Warehouse, Evolution Operations, Data Mart, Propagation Rules, Schema Versioning.

Abstract. Data Warehouse, accompanied with Online analytical processing, is considered as the core of the modern Decision support systems. The emergence of new analytical requirements and changes in organization business processes push the underlying information sources, destined to feed the data warehouse, to modify not only their data, but also their structure. This, obviously, has a direct impact on Data Warehouse and its associated Data Marts. Maintaining Data Warehouse structure becomes, therefore, a must; however, it is not sufficient. In fact, evolutions performed on the Data Warehouse schema have to be propagated on the related Data Marts in order to minimize costs, time-consuming and to guarantee the coherence of provided analysis results; this presents our first vision issue for which, we aim to provide an adequate solution. Another issue, which is as important as the precedent one, focuses on modeling a continuous temporal evolution phenomenon and therefore reducing inconsistent Online analytical processing queries results. Indeed, data returned by queries can be the result of an evolution phenomenon continued in several time intervals. Therefore, we nominate the versioning approach as a solution to keep traces of Data Warehouse / Data Mart schemas' modifications. Solving these two issues presents the key of organization Decision support systems durability and its material prosperity.

1 INTRODUCTION

As its data are often scattered and unstructured for analysis, the operational information systems seem inadequate for decision making. Toward this inadequacy, the Data Warehouse (DW) technology has emerged to collect and restructure data with the aim to be the process of a good decision making. Due to the continual evolving of decision makers' needs and the emergence of new business processes over time, organization operational system will be affected and therefore must include new data. To analyze these latter, their integration into the Decision Support Systems (DSS) becomes necessary. This, obviously, has a direct impact on DW. So, it must be renewable and adaptable to all changes that may occur; however, this is not sufficient to ensure the coherence of query results. Indeed, the strong dependence between the DSS components reveals the necessity to propagate the changes performed on DW towards its Data Marts (DMs). To achieve this propagation, we define a set of "if-then" type rules in order to identify the evolution operations that can affect DMs following DW structure evolutions. In the literature, works dealing with the DW evolution problem can be classified into three different approaches namely schema evolution, schema versioning and View Maintenance. Our goal is to express the improvement of DW schema over time. So, we proposed a solution based on the versioning approach as it keeps traces of schemas changes through time.

In this paper, we discussed two important issues: the first one focuses on how to propagate DW schema evolution towards DM; the second issue expresses how the versioning approach can be the best solution to guarantee consistency and coherence of Online Analytical Processing (OLAP) queries results.

This position paper is organized as follows. In Section 2, we provide an overview of related works to the DW evolution. Section 3 presents motivations and our position. Section 4 describes our proposed evolution approach. In section 5, we express our proposed approach efficiency by applying a set of propagation rules illustrated by a medical case study. Section 6 compares our proposed approach to another existing one. We conclude the paper in section 7.

Azaiez N. and Akaichi J..
How to Guarantee Analysis Results Coherence after Data Warehouse Schema Changes Propagation towards Data Marts?.
DOI: 10.5220/0005158304280435
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2014), pages 428-435
ISBN: 978-989-758-049-9

2 STATE OF THE ART

Solving DW schema evolution problems is a computational challenge in the midst of the continuous growth of technologies. Indeed, managing correctly all type of changes, which affect an organization DSS, can reflect the organization real world and therefore, guarantee its durability.

The literature is interested in the evolution problem and proposes solutions leading to better decisions making. These solutions are based on three approaches: *Schema evolution, schema versioning* and *maintenance of materialized views*.

2.1 Schema Evolution Approach

This approach is based on the assumption that the DW schema has only one version, the current one. Changes, which can affect DW schema, are translated into evolution operations updating the schema structure and the associated instances.

Following the study of a sample of works related to the DW schema evolution problem, we present a comparative study based on a set of relevant criteria.

Table 1: Comparative study between existing works based on schema evolution approach.

		(Hurtado *et al.*, 1999)	(Papastefanatos *et al.*, 2009)	(Taktak *et al.*, 2012)	(Azaiez *et al.*, 2013)
DW schema evolution	Dimensions	✔		✔	✔
	Hierarchies	✔		✔	✔
	Fact tables			✔	✔
Instances evolution		✔			
Materialized views evolution		✔			
ETL evolution			✔		
Evolution impact DW/DMs				✔	✔
Conformity to Meta-model				✔	
Prototype			HECATA-EUS		DWEv

Most of presented works treat different evolution operations related to various DW components such as dimensions and hierarchies. Some authors studied the effect of evolution operations on instances and materialized views (Hurtado et al., 1999), or even on the process of ETL (Papastefanatos et al., 2009). Others confirmed that the evolution problem can't be completely resolved only with the changes' full implementation on the meta-model level. That's why; they deepened their research to investigate conformity of DW schema to its meta-model (Taktak et al., 2012).

In general, the changes applied on DW schema and on its DMs are manual; Taktak et al. (2012) and Azaiez et al. (2013) resolved this gap. In fact, they proposed approaches capable to identify the impact of the DW evolutions on associated DMs.

However, the common drawback, that gathers all these works, is the impossibility to model a continuous temporal evolution phenomenon since the principle of the schema evolution approach avoid to keep the previous evolutions history.

2.2 Schema Versioning Approach

Unlike the principle of updating schema (schema evolution), the temporal modeling, on which the second approach is based, designed to keep traces of different DW changes in several versions; this is what is called *schema versioning approach.* Authors' works follow two different ways: either the dimension members historization or the full DW schema historization.

Table 2 summarizes some works classified according to some identified criteria in the context of schema versioning.

Table 2: Comparative study between existing works based on schema versioning approach (Zouari et al., 2008).

		(Eder *et al.*, 2001)	(Body *et al.*, 2002 ; 2003)
DW complete schema evolution operations	Dimension insertion/deletion		✔
	Level Insertion/deletion		✔
	Hierarchy insertion/deletion		✔
	Measure insertion/deletion		✔
	Fact insertion/deletion		
Dimension members evolution operations	Dimension members insertion/deletion	✔	✔
	Update of low attribute of dimension members	✔	✔
	Update of dimension members key		
	Dimension members subdivision/Fusion	✔	✔

After examining works presented recently, it seems that some authors were interested only in the evolution of dimension members, others were interested in operations affecting the full DW diagram. Obviously, those latter were also interested in keeping the history of dimension members since they treat the full DW diagram evolution problem.

We note that authors cited above neglected studying the evolution of the most dynamic part in a multidimensional schema; it is the Fact compound updates. Besides, alterations, affecting DW schema and its related DMs, are manual. Therefore, proposed approaches are limited to study what it must be evolved and neglect how to evolve it; this gap

deserves to be studied to guarantee a reliable organization DSS.

2.3 Materialized Views Maintenance Approach

The third evolution approach, called *maintenance of materialized views*, considers a DW as a set of materialized views constructed from data sources. This approach focuses on maintaining materialized views in response to data changes or to data sources changes and even to oversee the DW quality under schema evolution. Research works, related to view maintenance, can be classified in two categories:

- *View adaptation*: this approach consists in adapting views to changes by including metadata containing structural views updates.
- *View synchronization*: this approach consists in determining legal rewritings for affected views.

Table 3: Comparative study of works related to materialized views maintenance.

	Gupta et al., 1995	Bellahsene 2002	Akaichi et al., 2008	Quix 2005
View adaptation	✔	✔		
View synchronisation		✔	✔	
DWQ				✔

The DW administrator can bring modifications directly to views independently of data sources; that is called *view adaptation*. Furthermore, data sources can change their schema; this leads to lose of the coherence of materialized views. In this case, preserving the DW structural consistency becomes a must; this is called *structural view maintenance.*

Through the classification presented in the table above, we note that the *View adaptation approach* and *the View synchronization approach* are the focus points of the majority of the presented works. Despite the great changes' impacts on DW quality (DWQ), this gap was only treated in (Quix, 2005).

3 MOTIVATIONS AND POSITION

The DW technology was developed to integrate heterogeneous information sources for analysis purposes. Therefore, a DW is always renewable following changes that may affect its structure. These changes can be the translation of organizational business processes progressing over time, the evolving needs of decision makers that lead to DW structure enrichment with additional analyses axes, or even of removing of decision makers' needs vagueness occurred during the DW design stage.

According to related works discussed in section 2, the classification of approaches depends on the DW schema definition. In fact, the DW can be defined as a multidimensional schema (i.e. star or snowflake schema) or as a set of materialized views. For the multidimensional modeling, DW evolution approaches cover both of the schema evolution approach and schema versioning approach. However, for the materialized views modeling, the DW evolution approach includes the view adaptation and synchronization.

In general, Data Warehousing is extremely correlated to multidimensionality. Practically, the concept "Data warehouse schema" orients designers to think about "multidimensional modelling" more than "materialized modelling". Therefore, for maintaining DW after schema change it is required to choose one of the two schema approaches: the schema evolution approach and the schema versioning approach. We compared the two approaches and we found that the schema versioning approach is more adequate than schema evolution approach for DW schema maintenance. Indeed, the schema evolution approach consists on updating the old schema and keeping only the last schema version; this leads to lose data over time and consequently the impossibility to model a continuous temporal evolution phenomenon. On the contrary, thanks to the functionalities offered by the schema versioning approach, the evolutions history of all schema versions are kept; this may resolve the problem of queries which responses are returned over several time intervals. Another advantage of the versioning approach is manifested in predicting the impacts of future evolutions on the organisation development. This prediction can't be correctly derived only if it is based on the light of the previous decisions making.

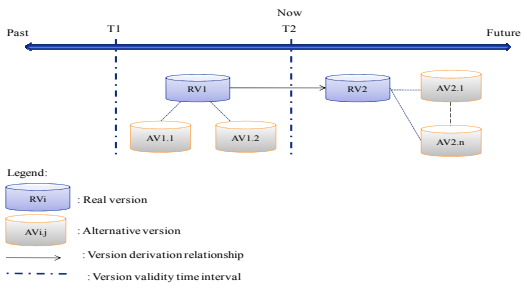

Figure 1: Versioning approach architecture (Oueslati et al., 2011).

Figure 1 presents the versioning approach archi

tecture. This approach aims to give the birth of a set of DW versions in time. Some are called real versions (*RV*), whose role is to reflect changes occurred in the real world, and others are called alternative versions (*AV*) which present virtual business scenarios.

RVi is a DW real version with $i \in \{1..., n\}$, *AVi.j* is a child of *RVi*; it's an alternative version and *RVi*→*AVi.j* means that *AVi.j* is a subset of *RVi*.

Whatever the kind of the version, this latter has its own time validity. The valid time interval of *RVi* is designed by *VTi [tb(RVi), te(RVi)]* as well as the valid time interval of *AVi.j* is designed by *VTi.j [tb(AVi.j), te(AVi.j)]*.

This paper proposes an approach that ensures the coherence and consistency of analysis results and consequently the organization durability and material prosperity. This requires relying on an reliable DSS that ensures the automation of evolution process tasks, and historization of previous evolutions.

4 APPROACH OVERVIEW

Organization business process evolution leads to the emergence of new data that must be analyzed for decision making. Consequently, their integration into the DSS becomes necessary in order to be analyzed. So, we propose to translate alterations into a set of comprehensive evolution operations to be applied on DW and therefore automatically propagated towards the associated DMs. Besides, our approach offers the possibility to keep traces of previous occurred evolutions in several versions.

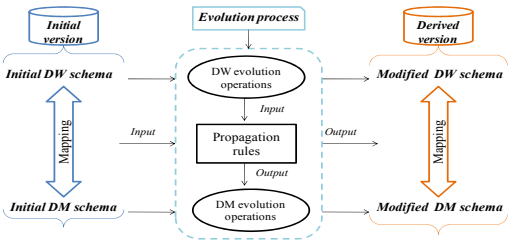

Figure 2: Proposed evolution approach architecture.

The overall proposed approach architecture is described in figure 2.

The proposed approach evolution process is composed of three steps:

- Identifying operations that can affect an initial DW schema to give the birth of several new DW versions (modified DW schemas); they are the DW derived versions.
- Defining a set of "if_then" type rules in order to identify updates affecting associated DM schemas. These rules take as input the type of the evolution operations (addition, deletion) applied on the initial DW model, the affected elements (table, column) and the various mappings DW/DMs. As output, rules give back operations which must be applied on DM schemas. This step presents the core of the evolution process.
- Applying the generated DM evolution operations on an initial DM schema gives the birth of several new DM versions (modified DM schemas); they are the DM derived versions.

Our approach advantage is that the horizontal evolution, covering multiple born DW/DMs versions, offers the possibility to model a continuous temporal evolution phenomenon since it keeps traces of different changes affecting the organization DSS overtime.

5 PROPAGATION RULES: MEDICAL CASE STUDY

To illustrate the different cases of changes occurred on DW schema and their impacts on related DMs, we rely on a DW relational model. Figure 3 shows a medical DW example constructed from tables which are interconnected with constraints. Figure 4 is a DM star schema called "Analyzing patient consultation" that we built from the medical DW of Figure 3.

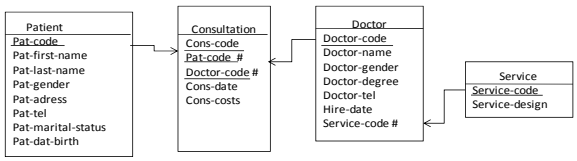

Figure 3: A medical DW example.

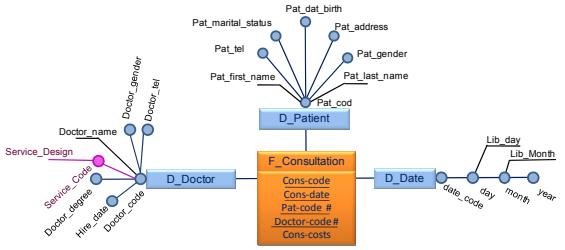

Figure 4: "Analyzing patient consultation" DM built from the DW of Figure 3.

To define rules that ensure the changes propagation, we use the following notations:

- T_D: A table T of DW which feeds a dimension D,

- T_F: A table T of DW which feeds a fact F,
- T_H: A table T of DW which feeds a hierarchy H,
- T_{id}: The identifier of a table T.

5.1 Addition Propagation Rules

In this section, we are inspired from our work in (Azaiez et al., 2013) which is based on schema evolution approach principles. However, the current work takes another way that leads to more effective analysis results. Indeed, we ameliorate the few rules proposed in the previous work, define other propagation rules that include evolution operations which seem important and deserve to be studied, and we illustrate all of them by a medical case study according to the schema versioning approach principles.

In this section, we define the addition propagation rules for two cases: addition of a new table and addition of a new column to the DW.

5.1.1 Table Addition

The addition of a new table T to DW can feed a new fact F, a new dimension D or a new hierarchy H.

We define a rule for each case.

- *Rule $R_{AT}1$. T* addition can create a new fact F

If T references several tables loading different dimensions (T_{D1}, T_{D2}...T_{Dn}) of DM, if T is not referenced by any table of the DW, if T contains one or several additive column(s) and if the primary key of T contains foreign key(s), then T can create a new fact F. Consequently, the dimensions of F are T_{D1}, T_{D2}...T_{Dn}. For example, it is proposed to add the table ***Hospitalisation*** (*Hospital-code, Pat-code#, Arrival-date, Release-date*) to DW. ***Hospitalisation*** contains one additive column which focuses on the nights number and generated from the formula (*Release date - Arrival date*). The table ***Hospitalisation*** refers to the table ***Patient*** that feeds the ***D_Patient*** dimension and it is not referenced by any table of the DW. According to the rule $R_{AT}1$, the table ***Hospitalisation*** can create a new fact conventionally called ***F_Hospitalisation*** and consequently the emergence of a new DM star schema related to the DW*;* ***D_Patient*** is a dimension ***F_Hospitalisation***.

- *Rule $R_{AT}2$. T* addition can create a new dimension D

If T is referenced by T_F that feeds a fact F, if T_{id} is atomic and if T contains columns that can be dimensional attributes (strong or weak), then T feeds a new dimension D for F. Suppose that we add the table ***Room*** *(Room-code, Room-category, one-night-price)* to the DW by connecting it to the table ***Hospitalisation*** that feeds the fact ***F_Hospitalisation.*** As the ***Room*** table contains an atomic identifier (*Room-code*), a column that may become a parameter (*Room-category*) and a column that can become weak attribute (*one-night-price*), it satisfies the rule $R_{AT}2$. Therefore, it transforms into a new dimension ***D_Room*** for the fact ***F_Hospitalisation.***

- *Rule $R_{AT}3$. T* addition can create a new hierarchy H

If T is referenced by T_D which feeds a dimension D, if T doesn't refer any table, if T_{id} is atomic and if T doesn't contain additive column(s), then T completes the dimension D with a hierarchy H by connecting D_{id} to the attribute T_{id}. Potential weak attributes of T_{id} parameter are the textual attributes of T. For example, the addition of the table ***Disease*** *(Disease-code, Disease-design)* which is referenced by ***Patient*** (feeds the dimension ***D_Patient*****),** completes the dimension ***D_Patient*** with a new hierarchy ***H_Disease*** (*Pat-code, Disease-code*). As *Disease-design* is a textual attribute, it is considered as a weak attribute of *Disease-code*.

We propose to express the principle of schema versioning approach in the case of a new table T is added to the DW model. We suppose that the basic DW version is the model presented in figure 3; it's the *RV1* of the DW model. We propose to add new DW versions following the applying of $R_{AT}1$, $R_{AT}2$ and $R_{AT}3$ rules. In general, the designer must choose to create a new alternative version (*AV*) either following another *AV* or following a Real version (*RV*); however, in the case of $R_{AT}1$, relying on the first solution is a must. Indeed, the $R_{AT}1$ output is to create a new fact table ***F_Hospitalisation*** following the addition of the table ***Hospitalisation*** to the DW schema; this requires the creation of a new *AV1.1*. Then, we propose to apply the rule $R_{AT}2$. The $R_{AT}2$ output is to create a new dimension ***D_Room*** following the addition of the table ***Room*** to the DW schema, and this new dimension can't be approved only if the table ***Room*** is related to a T_F in DW model. That's why, in order to reveal the evolution occurred following the application of $R_{AT}2$, we are obliged to create another *AV1.2* sequentially following the *AV1.1* which shows the evolution happened after applying $R_{AT}1$. The rule $R_{AT}3$ can be applied with different manners. Indeed, in this case, the designer isn't obliged to create a new *AV* sequentially following the previous one; the new *AV* can follow the *RV* since applying $R_{AT}3$ gives the birth of a new hierarchy whose related dimension have already existed in the *RV1*. In order to express evolutions

occurred due to applying the three rules in the same schema, we choose to give the birth a new *AV1.3*, that contains the $R_{AT}3$ applying results, sequentially following the previous one. As consequence, figure 5 presents a part of *AV1.3* containing *AV1.2* enriched with a new table ***Disease***. Then, a new DM version related to the *AV1.3* is created (Figure 6). It is a star schema which consists of the fact table ***F_Hospitalisation*** referencing a set of dimensions including the new dimension ***D_Room*** and the new hierarchy ***H_Disease***.

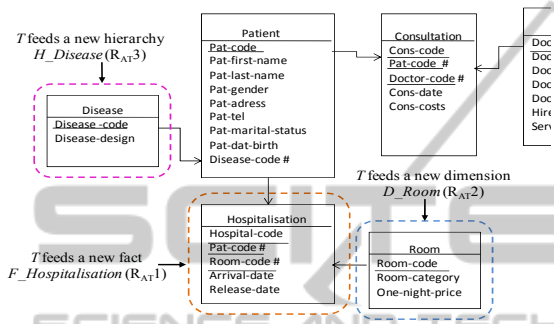

Figure 5: AV1.3: Addition of Hospitalisation, Room and Disease tables to the DW.

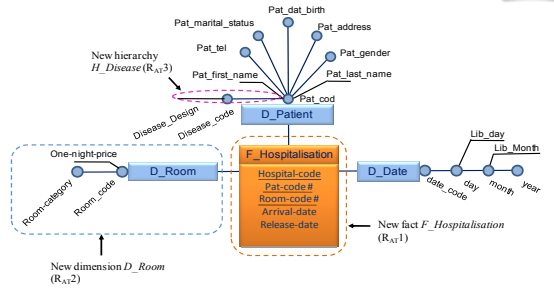

Figure 6: New Star schema version: "Analyzing patient hospitalization" built from AV1.3.

5.1.2 Column Addition

The addition of a column *C* to an existing DW can enrich it with a new measure *M*, an existing *D*imension with a new *A*ttribute *DA* or a fact with a new dimension *D*.

We define a rule for each case.

- *Rule* $R_{AC}1$. *C* addition can create a new measure *M*

If the column *C* is additive and is added to a table T_F, then *C* presents a measure for *F* in the DM. For example, it is proposed to add the column *Total-nights-costs* to the table ***Hospitalisation*** that feeds the fact ***F_Hospitalisation***. As the column *Total-nights-costs* satisfies the rule $R_{AC}1$, it feeds a new measure called *Total-nights-costs* for the fact ***F_Hospitalisation***.

- *Rule* $R_{AC}2$. *C* addition can create a new dimension attribute *DA*

If *C* is added to a table T_D and if *C* is non-additive, then *C* is considered as an attribute for the dimension *D* in the DM. The choice of the attribute role (weak or strong) is decided by the designer. For example, we suppose to add the non additive column *Room-loc* to the table ***Room*** that feeds the dimension ***D_Room***. The column *Room-loc* satisfies the rule $R_{AC}2$, so it feeds an attribute (strong) for the dimension ***D_Room***.

- *Rule* $R_{AC}3$. *C* addition can create a new dimension *D*

If we add *C* of date type to T_F, and if T_F doesn't contain columns of date type, then *C* can feed a temporal dimension in the related DM of *F*. For example, we propose to add the column *Date* to the table ***Consultation*** (Figure 3); however, it contains a column of date type. Therefore, the addition of the *Date* column does not have any effect neither on DW nor on DM as it doesn't satisfies the rule $R_{AC}3$ conditions. Moreover, we propose to add the table ***Laboratory_Exam*** (*Exam-code, Pat-code#, Exam-type, Exam-costs*) to the DW. The table ***Exam*** satisfies the conditions of $R_{AT}1$. This leads to the creation of a new fact so called ***F_Laboratory_Exam*** and, consequently, the emergence of a new DM called "*Analyzing patient laboratory exam"* related to the DW model*;* ***D_Patient*** is a dimension connected to ***F_Laboratory_Exam***. This latter doesn't contain columns of date type. According to the rule $R_{AC}3$, if we add a column *Exam-date* to the table ***Laboratory_Exam***, then the column *Exam-date* enriches the DM containing the fact ***F_Laboratory_Exam*** by ***D_Date*** dimension.

5.2 Deletion Propagation Rules

In this section, we define the deletion propagation rules for two cases: deletion of an existing table and the deletion of an existing column from the DW.

5.2.1 Table Deletion

The deletion of an existing table *T* from DW can lead to the elimination of an existing fact table *F*, an existing dimension table *D* or even a hierarchy *H*.

We define a rule for each case.

- *Rule* $R_{DT}1$. *T* deletion can eliminate an existing fact *F*

If the table T_F is eliminated from DW, then the fact table *F* will be automatically eliminated as well as the DM containing *F*. This deletion doesn't lead to the deletion of all dimensions; indeed shared dimensions will be maintained. In our example, if we delete the ***Laboratory_Exam*** table, then ***F_Laboratory_Exam*** will be eliminated as well as the DM "*Analyzing patient laboratory exam"*. ***D_Patient*** dimension which is a common dimension for ***F_Laboratory_Exam*** and ***F_Hospitalisation,*** will be maintained; any other specific dimensions, such as ***D_Date***, will be eliminated.

- *Rule* $R_{DT}2$. *T* deletion can eliminate an existing dimension *D*

If the table T_D is eliminated from DW, then the dimension table *D* will be eliminated from all DMs containing it. In our example, as the ***Patient*** table feeds only one dimension, its elimination from DW schema leads to the deletion of ***D_Patient*** from all DMs.

- *Rule* $R_{DT}3$. *T* deletion can eliminate an existing hierarchy *H*

If the table T_H is eliminated from DW, then the hierarchy *H* will be eliminated from the dimension *D* which contains it. In our example, if we eliminate the table ***Disease*** which enriches the dimension ***D_Patient*** with the hierarchy ***H_Disease***, then this latter will be deleted from every DM contains the dimension ***D_Patient.***

5.2.2 Column Deletion

The deletion of an existing column *C* from a DW table can lead to the elimination of an existing parameter *P*, an existing hierarchy *H*, an existing measure *M* or even an existing dimension *D*.

We define a rule for each case.

- *Rule* $R_{DC}1$. *C* deletion can eliminate an existing parameter *P*

If we delete a non additive column *C* which is not a primary key from a table T_D of DW schema, then its corresponding element in DM will be deleted from the dimension *D* which is fed by T_D. Thus, in our current example, the column *Pat-marital-status* of the table ***Patient*** is not a primary key and it is not an additive attribute. So, its deletion leads to the elimination of the corresponding parameter *Pat-marital-status* from the dimension ***D_Patient.***

If the column *C* of a table T_D supports a referential constraint toward another table T'_D of the DW, then *C* is a hierarchical level for *D*. Two cases can arise depending on whether or not T'_D refers to another table. In the following rule, we restrict our study to the case that T'_D doesn't refer to another table (end of hierarchy). In this context, we define the $R_{DC}2$ rule.

- *Rule* $R_{DC}2$. *C* deletion can eliminate an existing hierarchy *H*

The removal of *C* which satisfies conditions cited above, lead to the deletion of the hierarchy $TD_{Id} \rightarrow TD'_{Id}$ from DM(s) that contains it. As an illustration, if we eliminate the *Service-code* column from the table ***Doctor*** (Figure 3) which feeds the ***D_Doctor*** dimension (Figure 4), then the corresponding parameter of *Service-code* column which presents a hierarchy level in ***D_Doctor*** as well as its weak parameter *Service-design,* will be deleted.

- *Rule* $R_{DC}3$. *C* deletion can eliminate an existing measure *M*

If a numeric additive column *C* is removed from a table T_F of the DW, then the measure *M* which corresponds to *C* will be eliminated from *F*. In our example, if the column *Total-night-costs* is deleted from the table ***Hospitalisation*****,** then the measure called *Total-night-costs* will be deleted from ***F_Hospitalisation***.

- *Rule* $R_{DC}4$. *C* deletion can eliminate an existing dimension *D*

If the column *C* supports a referential constraint toward a table T_D and if it is deleted from T_F, then the *D* dimension will be deleted from the DM which contains *F*. In our example, the column *Patient-code* presents a foreign key in the table ***Hospitalisation*** which feeds the fact ***F_Hospitalisation*** in DM. So, its removal leads to the elimination of the dimension ***D_Patient*** from the DM called **"***Analyzing patient hospitalization*".

6 COMPARATIVE STUDY

Our approach is proposed to solve some gaps of previous works. The idea is to exploit versioning approach functionalities to keep traces of occurred changes propagated from DW towards DMs; this leads to coherent analysis results. Concerning evolution operations, we focus on applying propagation rules for two cases: tables and column addition and deletion to DW. On the contrary, in (Azaiez et al.,

2013), we investigated the problem of DW evolution only in the case of tables and columns addition. Besides, we chose the schema evolution approach as the base of the work.

The comparative study express that the current proposed approach offers coherent analysis results unlike results given in (Azaiez et al., 2013). In fact, in this latter, queries are unable to return data which are the results of an evolution phenomenon continued in several time intervals, since the schema evolution approach is based on the hypothesis that the DW schema has only one version; it's the current one.

The following table compares of our proposed approach versus the previous one:

Table 4: Comparative study.

		(Azaiez et al., 2013)	Our approach
Evolution approach	Schema Evolution	✔	
	Schema Versioning		✔
Evolution operations	Addition tables/columns	✔	✔
	Deletion tables/columns		✔

7 CONCLUSION AND PERSPECTIVE

In this paper, we presented an overview on the DW evolution problems. Indeed, we exposed some solutions proposed by different authors in recent years. To overcome the problem related of the DW schema changes and their impacts on DMs, we proposed an approach which deals with the propagation problem of DW changes on its DMs; this approach is based on "if-then" type rules. However, this is not enough to ensure the analysis results coherence and consistency. Therefore, we relied on the schema versioning approach to keep trace of evolutions affecting DW model and their impacts on related DMs.

This paper is limited at studying the evolution modeling of classic DWs that includes data which concerned only fixed objects, and neglected moving objects activities that generate a new data type so called "trajectory data"; those latter are stored in a mobile data central repository that called Trajectory Data Warehouse (TDW). As perspective, we propose to deal with the TDW evolution problems taking into account its new data type and structure changes.

REFERENCES

Akaichi, J., Oueslati, W., 2008. MAVIE: A Mobile Agents View synchronization system. In *first international conference on the applications of digital information and web technology* (pp. 145-150).Ostravem.

Azaiez, N., Taktak, S., Feki, J., 2013. DWEV : Un prototype pour l'évolution partielle du schéma multidimensionnel. In *7éme édition de la Conférence Maghrébine sur les Avancées des systèmes décisionnels (ASD*), Marrakech, Maroc.

Bellahsene, Z., 2002. Schema Evolution in Data Warehouses. *Journal of Knowledge and Information Systems*, 4 (3) (pp. 283-304).

Body, M., Miquel M., Bédard, Y., Tchounikine, A., 2003. Handling Evolutions in Multidimensional Structures. In *IEEE 19th International Conference on Data Engineering (ICDE)* (pp. 581-591). Bangalore, India.

Body, M., Miquel, M., Bédard, Y., Tchounikine, A., 2002. A multidimensional and multiversion structure for OLAP applications. In *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP* (pp. 1-6). McLean, Virginia, USA.

Eder, J., Koncilia, C., 2001. Changes of Dimension Data in Temporal Data Warehouses. In *Proceedings of the DaWaK'01 Conference*, (pp. 284-293). Munich, Germany.

Gupta, A., Mumick, I., Ross, K., 1995. Adapting Materialized Views after redefinitions. *SIGMOD 95*, (pp. 211-222).

Hurtado, C. A., Mendelzon, A. O., Vaisman, A. A., 1999. Maintaining Data Cubes under Dimension Updates. In *XVth International Conference on Data Engineering (ICDE 1999), IEEE Computer Society,* (pp.346–355).Sydney.

Papastefanatos, G., Vassiliadis, PP., Simitsis, A., Sellis, T., Vassiliou, Y., 2009. Rulebased Management of Schema Changes at ETL Sources. In *The International Workshop on Managing Evolution of Data Warehouses (MEDWa),* Riga, Latvia.

Quix, C., 2004. Repository Support for Data Warehouse Evolution. In *Proceedings of the International Workshop DMDW,* Heidelberg, Germany.

Taktak, S., Feki, J., 2012. Toward Propagating the Evolution of Data Warehouse on Data Marts. *In: MEDI 2012. Lecture Notes in Computer Science, Vol. 7602, Springer Verlag, Berlin Heidelberg* (pp. 178–185). Poitiers, France.

Zouari, I., Ghozzi, F., Bouaziz, R., 2008. Impact de l'évolution de nomenclature sur le versionnement des entrepôts de données. *Ingénierie des Systèmes d'Information*, volume 13, (pp. 85-114).

Oueslati, W., Akaichi, J., 2011. A Multiversion Trajectory Data Warehouse to Handle Structure Changes, *International Journal of Database Theory and Application*, Vol. 4, No. 2. (pp. 35-50).