IMPACTS OF DATA WAREHOUSE EVOLUTION ON DATA MARTS

Jamel Feki, Said Taktak
Mir@cl Laboratory
University of Sfax Tunisia, P.O.Box 1088
{jamel.feki, said.taktak}@fsegs.rnu.tn

ABSTRACT

Decision Support Systems are built around a Data Warehouse (DW) and a set of dependent Data Mart (DM). The DW gathers transversal data issued from several business processes of an organization; these data are required for decisional purposes. A DM is a business process oriented and, therefore, contains an extract of the DW data dedicated to a business process. However, in practice, business processes may evolve as well as new ones may emerge. Consequently, the DW should evolve in order to reflect these business process evolutions. In addition, the propagation of the DW changes on its DMs is fundamental. On the other hand, it is manually done in practice. Hence, propagation is time-consuming, expensive and error-prone process: this raises a new pertinent research issue: the impact of the DW schema evolution on the set of its dependent DMs. This paper studies the impacts of the DW schema changes on the DMs in an attempt to prepare the automation of the changes propagation.

Keywords: Decision support system, Evolution, Propagation, Data warehouse, Data mart

1. INTRODUCTION

In a Decision Support System (DSS) data flow, we distinguish three cascading data levels namely the Transactional Information System (IS) level, the Data Warehouse (DW) level, and the Data Mart (DM) level (cf. Figure 1). In this flow, the DW is loaded with cleaned and transformed data coming from one or several IS sources: these data are required for decisional purposes giving a transversal view/perception of the organization business processes. In its turn, a DW feeds its dependent DMs with extracts from the DW data. Data extracted from the DW and loaded into a DM are related to a specific business process to be analyzed through OLAP (On Line Analytical Processing) features. A DM aims at answering easily, quickly and efficiently analytical needs requested by decision makers in order to evaluate their business processes. According to this logic (cf., figure 1), there is a strong data-dependency of DMs from their DW. This dependency raises a new evolution problem: the impact of the DW schema evolution on the set of its dependent DMs.

This situation resembles to the IS evolution problem which was addressed by several works and according to many viewpoints [1-3], but it is sensibly different. From the IS side, some research have focused on the evolution of the IS content in terms of stored data values, while others concentrated their efforts on the changes undergone by the structures of these data (i.e., data model or schema). In this context, some of the proposed approaches follow a conventional trend (i.e., not using a special technology) while, more recent others are based on the model driven engineering paradigm in order to harness the power of this emergent technology: gains in development time, costs, performances and sustainability.

\[\text{Figure 1. Modeling levels of the DSS}\]

Considering the importance of the evolution problem in the DSS domain, some recent works have been interested with it; nevertheless most of the proposals have resorted to a simple adaptation of the classical solutions (i.e., solutions proposed for IS) to the DW domain. We can classify these proposals into three main categories: Schema Evolution [4-5], Schema Versioning [6-7], and View Maintenance [8-9].

The first category, i.e., schema evolution, assumes the hypothesis of unique schema. It considers that the schema can have only one version at a time: the current version according to which all data are stored. Consequently, any schema changes will affect the current schema version that transforms into a new version. Furthermore, this operation is accompanied with the data migration from the old schema version to the new one. However, this migration requires writing - often manually - new appropriate migration code.
Contrary to the principle of evolution schema, the second category addresses the schema versioning which aims to keep track of changes made to different versions of the DW schema through time. Schema versioning is based on temporal extensions that explicitly incorporate the temporal dimension to the schema.

Finally, the third category assumes that a DW is a set of materialized views directly loaded from the IS data source. Note that a materialized view is a view that has a content computed and physically stored as a database table. Moreover, a materialized view is linked to its data source; this link allows a periodic refreshment of its content so that it contains recent data issued from the data source. Naturally, this refreshment concerns only the data and, therefore, does not affect the structure of the materialized view which is assumed static through time. However, in practice, changes affecting the data model of the source require manual maintenance of these materialized views; this lack constitutes a significant handicap for the materialized view problem.

Through these proposals, we can claim that all evolution strategies tackle a single modeling level at a time (i.e., DW or DM exclusively). Indeed, schemas before and after changes conform to a single meta-model. Contrary, in DSS we deal with two different models since the DW is often structured as a conventional database (e.g., relational) whereas DMs are designed according to the multidimensional model (cf. Figure 4). In fact, to the best of our knowledge, schema evolution problems which impact dependent schemas expressed in different models have not yet received their full part of investigation in the DW domain.

Our objective in this paper is to study the impact of the DW schema changes on its dependent DM schemas.

This paper is organized as follows. In section 2 we give an overview of evolution works in the DW domain. Section 3 discusses the related works, gives the motivations and introduces our proposal. In section 4 we census DW and DM evolution operations, and their connection. In section 5 we define propagation rules from a DW towards a DM and illustrate them. Finally, section 6 concludes the paper and enumerates our perspectives.

2. RELATED WORKS

All literature works related to DSS evolution have addressed either the DW or DMs separately. Thus, this section discusses evolution contributions situated at a single level.

2.1HURTADO ET AL.

Hurtado et al. [10] treated the evolution mainly in the multidimensional model (fact and dimension; cf. figure 4 that depicts the multidimensional concepts) for the star or snowflake schemas. They proposed an abstract model and a set of evolution operators to define the alterations applicable to dimensional schemas and their instances.

These operators modify a multidimensional schema by allowing the addition or removal of hierarchical levels. In addition, the authors studied the effect of these modifications on the materialized views and, proposed certain adaptations and an algorithm for view maintenance.

Note that this approach is purely theoretical [11]. Moreover, to the best of our knowledge, no prototype has been developed to validate the approach.

2.2 BLASCHKA ET AL.

In the BabelFish project, Blaschka et al. [5] proposed the FIESTA software tool for DW modeling; they also suggested evolution operations for dimensions and facts. These evolution operations are expressed by means of a conceptual graph called ME/R (Multidimensional Entity/Relationship model) to be later propagated to the relational or multidimensional model. To do so, the authors formally define the multidimensional schema as well as their different evolution operators. They classified these evolution operators into two subsets.

The first subset includes operators, presumed by the authors, without effect on the multidimensional DM schema and its instances; for example, inserting an attribute into a dimension without any link to existing hierarchies, or deleting a hierarchy level from a dimension disconnected from facts [12].

The second subset includes operators that affect the schema or its instances, for example, inserting a new fact table, removing a dimension or connecting an attribute to a level of a hierarchy. These operations are basic and, therefore, can be combined to satisfy more complex evolution needs.

Blaschka et al. works study the evolution of dimensions and their hierarchies as in [10]; in addition, they treat the changes affecting the fact table and adding of a new hierarchical level without positional constraint. The developed software tool supports a methodology for the evolution of multidimensional schemas allowing the move from the conceptual to the logical level. However, these operations are dedicated to the star schema only. Note that a star schema is limited to a single fact connecting several dimensions as in figure 4.

2.3 BENITEZ-GUERRERO ET AL.

In [13], Benitez-Guerrero et al. proposed a prototype called WHES (WareHouse Evolution System) for the creation and the evolution of DWs. In order to achieve this, the authors defined an evolution model and the MDL language (Multidimensional Data Language). Their evolution model includes a set of primitives for multidimensional schema evolution. This set allows the creation of new dimension and cube schemas, and the deletion and modification of existing multidimensional schemas.
Their MDL language offers schema evolution primitives. It represents a means of communication between the DW designer/administrator and decision makers.

All these evolution primitives are inspired from existing works [10], [5]. However, to ensure the consistency of the modified schema resulting from successive changes, the authors developed a set of complementary primitives qualified as High level.

2.4 FAVRE ET AL.

Favre et al. [14] based the DW evolution on the evolution of the decision makers’ analytical requirements. So, they propose a user-driven evolution model to gather the user’s knowledge and integrate them into the DW in order to offer new analytical perspectives (i.e., hierarchies).

The proposed model is scalable, formal and based on “If-Then” rules to alter existing hierarchies by inserting a granularity level (i.e., parameter) which provides users with new OLAP alternatives. A software tool baptized “WEDriK” supports the evolution model.

2.5 PAPASTEFANATOS ET AL.

Papastefanatos et al. [15-16] were interested in IS evolution. In fact, they studied the effect of the data source schema evolution on objects of the Data Base Management System (i.e., queries, materialized views, constraints, procedures, etc.). In [15], the authors discuss the impact of this evolution on the ETL (Extract-Transform-Load) process that feeds the DSS with data. Their HECATAEUS tool assists the DW/DM designer with a mechanism for adapting the ETL activities after data source changes.

This proposed approach is based on a technical representation that maps all the essential components of the ETL process and produces a graph model of evolution [16]. Following a change of the graph element, the tool detects the graph elements that have to be affected and then highlights the changes to be made according to predefined rules.

However, this approach was limited to the ETL process but did not address the impact of the IS evolution on the DW and its DM schemas.

3. DISCUSSION, MOTIVATIONS AND PROPOSAL

3.1 DISCUSSION

The overview of the related works concludes that the main contributions were interested in the evolution of the DW components separately: evolution of dimensions, evolution of facts, parameters... Furthermore, these evolutions are based on the evolution of decision makers’ requirements. In addition, some authors studied the effect of the evolution operations on the DW data instances and on materialized views [4] or on data cubes [13]. In Table 1, we compare the above studied approaches according to seven criteria we have established.

To the best of our knowledge, we can claim that the evolution issues were addressed at a single level: data source level or data mart level. In addition, the impact of this evolution on the DW was limited to the loading ETL process.

On the other hand, we can assert that in almost all cases, when a DW schema evolves then the evolution of its dependent DMs is manually done in practice. As an intrinsic consequence, the data migration from the old to the new schema is also a manual task and often necessitates writing the migration code. Writing this code requires first, good knowledge of the maneuver evolution and, secondly high programming skills in order to perform a safe migration of the data, i.e., without loose/perturbation of information.

Table 1. Comparison of DW evolution approaches [18].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema evolution</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dimension</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fact table</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Instances evolution</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Materialized views</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>evolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cubes evolution</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ETL evolution</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Conform to a Meta-model</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Software prototype</td>
<td>FIESTA</td>
<td>WES</td>
<td>WEDriK</td>
<td></td>
<td>HECATAEUS</td>
</tr>
</tbody>
</table>

As well, we would like to stress that none of the existing contributions in data warehousing has addressed the identification of the impact of DW schema changes on the DMs so far. In addition, How to propagate these DW changes on DMs has not been tackled yet.

Moreover, this issue has not been addressed neither manually nor semi-automatically towards an evolutionary DSS; i.e., a DSS capable to evolve semi-automatically by deriving a new DM schemas providing decision makers with new potential OLAP analyses.

3.2. MOTIVATIONS

In fact, the DW schema may change over time. Kimball [19] claims that it is difficult to definitively determine the schema of a DW during the design phase. Consequently, it is often necessary to modify this schema after its implementation. These changes can have multiple reasons, for example, i) evolving needs of decision makers, leading to the add of new topics to the DW or its enrichment with additional analyses axes (i.e., dimensions); ii) the incompleteness of needs initially captured during the design phase of the DW (case of Top down approaches): decision makers can express their needs precisely and completely only when they start using the warehouse; iii) changes in organization business processes over time, this evolution probably affects the working procedures and even the data models of their operational system.
In all these situations, this evolution merits to be propagated on the DMs. In fact, we can distinguish two types of cascading evolutions: schema evolution, and ETL evolution; being cascading, the first type implies the second one. We will be interested in the first evolution type.

3.3. PROPOSAL

In order to consider the DW schema evolution and to automate its propagation on the DMs, we propose the overall architecture described in Figure 2. Note that this architecture depicts two evolution types namely: Horizontal evolution and Vertical evolution.

**Figure 2. Proposed approach for propagating DW evolutions towards the DMs.**

Horizontal evolution accepts the DW schema and a set of evolution operations (such as add new table, add a column to table) and then produces a modified version of the input DW schema. This type is very close to the database evolution which is widely studied in the IS literature and, therefore, does not represent our research concerns.

Vertical evolution aims at deriving a set of DM evolution operations (such as adding or modifying a fact, dimension, parameter…) and executing these operations. This derivation is based on i) a given set of DW evolution operations already realized on the DW schema, and ii) a Generic propagation model (GPM). This GPM describes how to transform each DW change into DM change(s). It automatically captures the DW changes; identify which DMs components are concerned with these changes, and what is the effect of this change on the DMs components. To do so, a mapping model of the ETL process is necessary; it associates DW-components with DM-components. The definition of this GPM model is out of the scope of this paper.

The execution of the derived set of operations issued from the GPM transforms a DM schema into a new schema reflecting the changes operated on the DW. These transformations are said vertical according to the MDA terminology because they involve two different models: the DW model and the multidimensional model of DMs.

Since this paper studies the impact of the DW evolution on its dependent DMs, in the remaining subsections, we enumerate the changes affecting the DW schema components; also we detail the changes which may occur on the DM schema and then, we establish connections between the operations of the DW-evolution and those of the DM-evolution. Thus, we answer questions as “What DM-Component would derive from a created DW table?”, “What does a new column should derive? (e.g., fact measure, parameter)”…

4. DW AND DM EVOLUTIONS

4.1 DM SCHEMA EVOLUTION OPERATIONS

We classify the DW-schema evolution operations into two categories: Basic operations and Composite operations.

— Basic operations are among the DBMS data definition language commands used in implementing the DW schema. Mainly, they are addition, deletion and modification of attributes, tables and constraints (cf. Table 2).

— Composite operations are obtained by composition of basic operations. As an example, the decomposition of a DW table T into two tables T1 and T2 requires the following basic operations: create tables T1 and T2 each composed of a subset of columns issued from T, then drop table T, and finally modify the constraints.

Note that in Table 2, a dashed cell means that the operation is non basic because it is a combination of the two basic operations Add and Delete.

**Table 2. Basic DW evolution operations.**

<table>
<thead>
<tr>
<th>DW component</th>
<th>Attribute</th>
<th>Table</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Delete</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Modify</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

In order to illustrate the different evolution operations we consider the relational DW of Figure 3 and its multidimensional DM of Figure 4.

**Figure 3. A sample relational DW schema.**

4.2 DM SCHEMA EVOLUTION OPERATIONS

Independently of any evolution operation executed on the DW, all expected DM evolution operations are those of Table 3 which shows that we can Add New Fact with
measure, or a Factless fact (i.e., Fact without measures for event recording [19] as attendance of students to classrooms), Normal, Degenerate or Temporal dimensions. We can also Add New Hierarchy levels to an existing dimension… Concerning the Delete, we rarely withdraw a DM component such as a Measure, a Factless fact … Nevertheless in practice; we can provide for the DW administrator these maneuvers.

Table 3. DM evolution operations.

<table>
<thead>
<tr>
<th>DM Operation</th>
<th>Fact</th>
<th>Hierarchy</th>
<th>Onion Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add New</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Add To Existing</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Delete</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

So far, we have identified the evolution operations at two separated modeling levels: the DW level (cf. Table 2) and the DM level (cf. Table 3). As our objective is to propagate the DW evolutions on DMs, then we need to connect each operation of the first level with its corresponding operations in the second level. In other terms, we need to identify the impact of each DW evolution operation on the existing DM schemas.

4.3 CONNECTING DW-EVOLUTION WITH DM-EVOLUTION

Obviously, not all checked cells of the same row in Table 3 could be considered simultaneously. For instance, creating a new DW table cannot create measures and Factless fact at the same time. Therefore, we need to associate each DW evolution operation with the corresponding plausible DM evolution operation(s). This should be done in an abstract way, i.e., independent of any particular DW/DM schema instances. In this sense, Figure 5 shows what kind of DM components could be created after a table is added to the DW.

Furthermore, more precisely, the effect of creating a DW table on the DMs depends on the structure of the created table (does the table represent an entity to produce a dimension or an association to produce a fact?) [20]. Thus, a DW table may be used to create a new subject of analyses (new fact with or without measures), new contexts of analyses within for existing facts (new dimension: degenerated or with hierarchies), a new hierarchy… To identify the effect of each table (respectively, column, constraint) we require the definition of propagation heuristics. In the next section, we define some propagation rules.

5. PROPAGATION RULES

A table T added to the DW can play different roles: it can be used to define new context of analyses (i.e., dimension) within existing DMs, or to create a new subject of analyses (i.e., fact). We distinguish several cases according to whether table T is referenced, or T references other tables in the DW. In the following, we describe four propagation rules and we illustrate them using the DW schema of figure 3 and the DM schema of figure 4.

Rule R_F. Table T creates a new fact

If T represents a relationship between entities, i.e., i) T is not referenced by any table of the DW, and ii) T references several tables loading different dimensions of the DM and if the primary key of T contains foreign keys then T derives a new fact.

For example, the creation of the SCORE_PROD (#{Id_Prod, Id_Cust, Score,...}) table referencing PRODUCT and CUSTOMER tables which respectively feed dimensions D-PRODUCT and D-CUSTOMER may result in the creation of a new fact `F-SCORE_PROD` related to both dimensions CUSTOMER and PRODUCT. This new fact will have numeric attributes of SCORE_PROD as its plausible measures.

Note if T has numerical attributes then it derives a measured fact otherwise T derives a Factless fact.

Rule R_D. Table T derives a new dimension

If a table T added to the DW is referenced (via a foreign key constraint) by one table noted T’ (of the DW) that feeds a fact F, then T may feeds a new dimension called
**D-T** for **F**. The attributes of **D-T** are non-numerical attributes issued from **T**.

For example, the creation, in the DW, of the table **RETAIL_OUTLET(Id_Outlet, Name, City...)** referenced by the **SALE** table (an alteration of the SALE table is required in order to add the foreign key constraint) which feeds the fact **F-SALE**, leads to the creation of a new dimension called **D-RETAIL_OUTLET** linked to the **F-SALE** fact. Consequently, this expands the analyses of the **Sale_Amount** measure in order to be aggregated by Retail outlet too.

![Figure 6. Dimension derived from the Retail_Outlet table for the F-SALE fact.](image)

**Rule R_{TP}.** **Table T** appends a terminal parameter to an existing hierarchy

If a DW table **T** is i) referenced by **n** (n ≠ 1) DW tables **T_{1}{^D}... T_{n}{^D}** those feed dimensions with data, and ii) **T**, does not refer to any table, then **T** can feed a new level of hierarchy in each of these dimensions. The identifier of **T** will be a terminal parameter. Textual attributes of **T** become weak attributes for the added level.

For instance, when **n=1**, creating the table **CONTINENT(Id_Cnt, CName)** referenced by **COUNTRY** (add the foreign key constraint to **COUNTRY** table) which feeds the dimension **D-CUSTOMER** leads to the creation of a new terminal parameter within the **CUSTOMER** dimension. Thus, we extend the analyses of the **Sale_Amount** measure enabling to get aggregated values as the ‘Total of sales by continent’.

![Figure 7. Terminal parameter ‘Continent’ added to hierarchy of D-CUSTOMER dimension.](image)

**Rule R_{MP}.** **T** inserts a parameter in the middle of an existing hierarchy

![Figure 8. Table T referred by T_{1}{^D} and references T_{2}{^D}.](image)

If table **T** i) is referenced by a DW table **T_{1}{^D}** loading a dimension **D**, and ii) if **T** references a table **T_{2}{^D}** which loads the same dimension **D** (cf. figure 8), then **T** can feed a new hierarchy level in **D**. The identifier of **T** becomes a parameter inserted between the two parameters issued from **T_{1}{^D} and T_{2}{^D}** of **D**; textual attributes of **T** are candidate weak attributes for the added level.

As an illustration, creating the table **REGION (Id_Region, RName, #Id_Cntry)** referenced by the table **CITY and COUNTRY** table, leads to the insertion of the new parameter **RName**, or (Region) located between **City** and **Country** parameters. Figure 9 shows this result.

![Figure 9. Parameter ‘Region’ inserted in the middle.](image)

## 6. CONCLUSION

In this paper, we are interested in studying the impact of the data warehouse schema evolution on the data marts schemas. To reach our objective, we studied the related works in the evolution of DSS and then we have presented the recapitulation. This allowed us to point out several lacks. After that, we have classified the schema evolution operations into two categories: basic operations and composite ones. Basic operations are directly supported by the DDL (Data Definition Language) of the DBMS whereas others are combinations of these operations. For basic operations, we particularly focused on adding a table to the DW which is modeled as a relational database.

In order to propagate the effect of the table addition on DMs, we have examined the DM evolution operations and defined a set of propagation rules. These rules serve to identify which multidimensional components of the DM have to be affected and then derive evolved DM schemas.

Currently, we continue to study carefully the remaining operations on the DW and the various cases of plausible DM alterations. In parallel, a software prototype is under construction.

Our long term objective is to provide DSS designers/administrators with a reusable approach that ensures the evolution of DM schemas after a DW schema evolves. To do so, our approach aims to be MDA compliant, this is to benefit of this paradigm such as reducing time and cost by automating the generation of code.

## REFERENCES


