A Knowledge-driven Data Warehouse Model for Analysis Evolution

Cécile FAVRE [1], Fadila BENTAYEB and Omar BOUSSAID
ERIC Laboratory, University of Lyon 2, France

Abstract. A data warehouse is built by collecting data from external sources. Several changes on contents and structures can usually happen on these sources. Therefore, these changes have to be reflected in the data warehouse using schema updating or versioning. However a data warehouse has also to evolve according to new users' analysis needs. In this case, the evolution is rather driven by knowledge than by data. In this paper, we propose a Rule-based Data Warehouse (*R-DW*) model, in which rules enable the integration of users' knowledge in the data warehouse. The *R-DW* model is composed of two parts: one fixed part that contains a fact table related to its first level dimensions, and a second evolving part, defined by means of rules. These rules are used to dynamically create dimension hierarchies, allowing the analysis contexts evolution, according to an automatic and concurrent way. Our proposal provides flexibility to data warehouse's evolution by increasing users' interaction with the decision support system.

Keywords. Data Warehouse, Schema Evolution, Knowledge, Rule

Introduction

The design of integrated concurrent engineering platforms has received much attention, because competing firms strives for shorter design delays and lower costs. Concurrent engineering allows for parallel design, thus leads to shorter design to market delays. It however requires advanced coordination and integration capabilities [4]. Concurrent engineering, also known as simultaneous engineering, is a non-linear product or project design approach during which all phases operate at the same time.

Data warehouse systems must operate in a fully concurrent environment. Materialized views must be maintained within the data warehouse dynamically with the data provided from data sources. Data sources can generate concurrent data updates and schema changes within the data warehouse. However, we think that data warehousing is a technology that is difficult to consider from a concurrent engineering point of view concerning the users' implication in the process. The data warehouse design corresponds to a linear process in which users are indirectly implied. First, a study of the data sources and the users' analysis needs is needed. Second, the model of the data warehouse has to be built. Last the ETL (Extract Transform and Loading) process has to be defined and implemented. End users make analyses entirely driven by the data warehouse model. By considering a concurrent engineering approach for users in the data warehouse, we need to cope evolutions of analysis possibilities defined by users.

[1]Corresponding Author: Cécile Favre, ERIC Laboratory - University of Lyon 2, 5 av. Pierre Mendès-France, 69676 Bron Cedex, France; E-mail: cfavre@eric.univ-lyon2.fr.

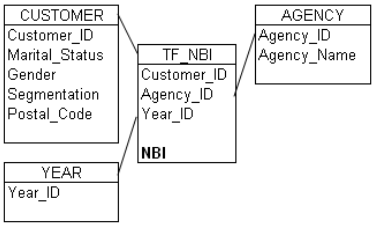

Figure 1. Data warehouse model for the NBI analysis.

It is impossible to take into account the evolution of all users' analysis needs, particularly when these needs are based on their knowledge. Indeed, sometimes users have knowledge which is not represented in the data warehouse but which is likely to be used for an analysis. To enable evolution of data warehouse, the administrator has to centralize the analysis needs and knowledge on which they are based. This process is a difficult task to achieve. Thus, we propose a new data warehouse model based on rules, named *R-DW* (*Rule-based Data Warehouse*), in which rules integrate users' knowledge allowing real time evolution of the analysis possibilities.

Our *R-DW* model is composed of two parts: a "fixed" part, defined extensionally, and an "evolving" part, defined intentionally with rules. The fixed part includes a fact table and dimensions of the first level (dimensions which have a direct link with the fact table). The evolving part includes rules which define new analysis axes based on existing dimensions and on new knowledge. These rules create new granularity levels in dimension hierarchies. Thus this model makes the evolution of analysis needs expressed directly by users easier, without the administrator's intervention.

The remainder of this paper is organized as follows. First, we present a motivating example in Section 1. Section 2 and Section 3 are devoted to the principles of our *R-DW* model and its formal framework. We also present an implementation and a running example with banking data in Section 4. Then we discuss in Section 5 the related work regarding schema evolution and flexibility offered by rule-based languages in data warehouses. We finally conclude this paper and discuss research perspectives in Section 6.

1. Motivating example

To illustrate our approach throughout this paper, we use a case study defined by Le Crédit Lyonnais french bank (LCL[2]). The annual Net Banking Income (NBI) is the profit obtained from the management of customers account. It is a measure that is studied according to dimensions customer (marital status, age...), agency and year (Figure 1).

Let us take the case of the students portfolio manager of LCL. This person knows that a few agencies manage only students accounts. But this knowledge is not visible in the model. It therefore cannot be used to carry out an analysis about student dedicated agencies. This knowledge represents a way to aggregate data, under the form of "if-then" rules, as the following rules which represent the knowledge on the agencies type:

[2]Collaboration with the Management of Rhône-Alpes Auvergne Exploitation of LCL-Le Crédit Lyonnais within the framework of an Industrial Convention of Formation by Research (CIFRE)

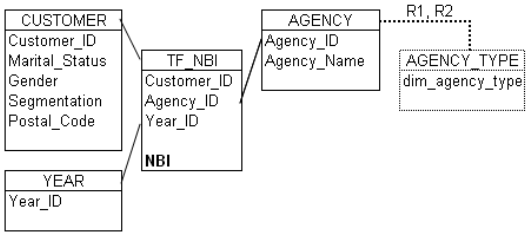

Figure 2. Rule-based Data Warehouse conceptual model for the NBI analysis.

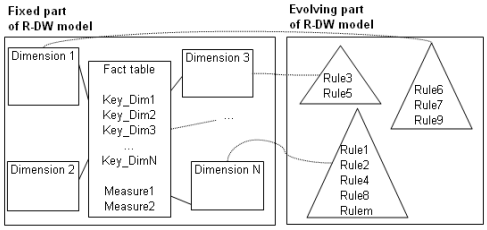

Figure 3. *R-DW* model.

$(R1)\ if\ Agency\_ID \in$ {'01903','01905','02256'} $then\ dim\_agency\_type =$ 'student'
$(R2)\ if\ Agency\_ID \notin$ {'01903','01905','02256'} $then\ dim\_agency\_type =$ 'classical'

The objective is then to carry out new analysis based on the user's knowledge. For an example, the objective is to build aggregates, by considering that the facts to aggregate concern a student agency (R1), or a classical agency (R2). Our objective is thus to integrate these rules into the model. Indeed, rules allow us to build the level `AGENCY_TYPE`, by defining the values of the `dim_agency_type` attribute in the dimension hierarchy (Figure 2). To achieve this objective, we propose the *R-DW* model.

2. The R-DW model

The *R-DW* model is composed of two parts: one fixed part, defined extensionally, and one evolving part, defined intentionally with rules (Figure 3). The fixed part can be seen as a star schema because it is composed of a fact table and first level dimensions. The evolving part is composed of rules which generate new granularity levels in dimension hierarchies based on users' knowledge and existing dimensions.

Our model provides to the users a way to express their rules to define dimensions hierarchies. It presents many advantages comparing to existing models by allowing: (1) to dynamically create hierarchies, (2)to make analysis on evolving contexts, (3) to increase the interaction between the user and the information system since the user can integrate his own knowledge.

In the *R-DW* model, rules, namely aggregation rules, are used to create dimension hierarchies by defining the aggregation link between two granularity levels in a dimension hierarchy. These rules are defined by users who express their knowledge. The aggregation rules are "if-then" rules. These rules have the advantage of being very intelligible

for users since they model information explicitly. The then-clause contains the definition of a higher granularity level. The if-clause contains conditions on the lower granularity levels. The following rules define the granularity level `AGENCY_TYPE` through the `dim_agency_type` attribute, basing on the `AGENCY` level (`Agency_ID` attribute):
$(R1)\ if\ Agency\_ID \in$ {'01903','01905','02256'} $then\ dim\_agency\_type =$ 'student'
$(R2)\ if\ Agency\_ID \notin$ {'01903','01905','02256'} $then\ dim\_agency\_type =$ 'classical'

The rules thus make it possible to integrate knowledge to define the various granularity levels of the dimension hierarchies. The advantage in building the dimension hierarchies with rules is that we are able to take into account the users' knowledge in real time. Therefore, the data warehouse model becomes more flexible for analysis. Indeed the users can analyze data according to the new granularity levels defined by their rules.

To take into account the knowledge or the analysis needs of different users simultaneously, we have to deal with "versions" of rules when the users define the same analysis needs by different ways. Let us take the example of age groups definition starting from the ages of the table `CUSTOMER`. The following classes can be defined by two users:

User 1	User 2
$if\ Age < 60\ then\ dim\_age =$ 'less than 60 years old'	$if\ Age < 18\ then\ dim\_age =$ 'minor'
$if\ Age \geq 60\ then\ dim\_age =$ 'more than 60 years old'	$if\ Age \geq 18\ then\ dim\_age =$ 'major'

3. Formal framework

We represent the Rule-based Data Warehouse model *R-DW* by the following triplet:

$$\textit{R-DW} = (\mathcal{F}, \mathcal{E}, \mathcal{U})$$

where $\mathcal{F}$ is the fixed part, $\mathcal{E}$ the evolving part et $\mathcal{U}$ the universe of the data warehouse *R-DW*.

Definition 1. *Universe of the data warehouse*
The *universe of the data warehouse* $\mathcal{U}$ is a set of attributes, such as:

$$\mathcal{U} = \{B_1, \ldots, B_\alpha, \ldots, B_z, C_1, \ldots, C_\beta, \ldots\} = \{B_\alpha, 1 \leq \alpha \leq z\} \cup \{C_\beta, \beta \geq 1\}$$

where $\{B_\alpha, 1 \leq \alpha \leq z\}$ is the set of z predefined attributes (in the fixed part $\mathcal{F}$) and $\{C_\beta, \beta \geq 1\}$ is the set of generated attributes (defined in the evolving part $\mathcal{E}$).

Definition 2. *Fixed part of R-DW*
The *fixed part of R-DW* is represented by:

$$\mathcal{F} = < F, \mathcal{D} >$$

where F is a fact table and $\mathcal{D} = \{D_s, 1 \leq s \leq t\}$ is the set of t first level dimensions which have a direct link with fact table F. We assume that these dimensions are independent.

Example 2. In the Figure 1, $\mathcal{F} =<TF\_NBI, \{AGENCY, YEAR, CUSTOMER\}>$ is the fixed part of the *R-DW* for the NBI analysis.

The expression of new analysis needs induces the definition of new granularity levels in dimension hierarchies.

Definition 3. *Dimension hierarchy and granularity level*
Let *R-DW* $= (< F, \mathcal{D} >, \mathcal{E}, \mathcal{U})$ be a data warehouse.

Let $D_s.H_k, k \geq 1$ be a *dimension hierarchy* $D_s \in \mathcal{D}$.

The dimension hierarchy $D_s.H_k$ is composed of a set of w ordered granularity levels noted L_i :
$D_s.H_k = \{L_1, L_2, \ldots, L_i, \ldots, L_w, w \geq 1\}$ where $L_1 \prec L_2 \prec \ldots \prec L_i \prec \ldots \prec L_w$.

The granularity level L_i of the hierarchy H_k of dimension D_s is noted $D_s.H_k.L_i$ or L_i^{sk}. The granularity levels are defined with attributes called generated attributes.

Definition 4. *Generated attribute*

An attribute $C_\beta \in \mathcal{U}, \beta \geq 1$, is called *generated attribute*.

C_β characterizes a granularity level in a dimension hierarchy. To simplify, we suppose that each granularity level of dimension hierarchy is represented by only one generated attribute, even if it is possible to generate more than one attribute per level.

Thus, the generated attribute C_β which characterized the granularity level L_i of hierarchy H_k of dimension D_s is noted $L_i^{sk}.A$. The values of these generated attributes are defined by using the evolving part of *R-DW*.

Definition 5. *Evolving part of R-DW*

The *evolving part of R-DW* is represented by

$$\mathcal{E} = \{< \mathcal{R}_{i.}, L_i^{sk}.A >\}$$

where $\mathcal{R}_{i.} = \{r_{ij}, 1 \leq i \leq w, 1 \leq j \leq v\}$ is a set of v aggregation rules defining the values of the generated attribute $L_i^{sk}.A$. $\mathcal{E}$ represents the set of w granularity levels of hierarchy H_k of dimension D_s and their associated rules.

Definition 6. *Aggregation rule*

An *aggregation rule* defines the aggregation link which exists between two granularity levels in a dimension hierarchy. It is based on a set $\mathcal{T}$ of n rule terms noted RT_p, such as:

$$\mathcal{T} = \{RT_p, 1 \leq p \leq n\} = \{U \; op \; \{set|val\}\}$$

where U is an attribute of the universe $\mathcal{U}$; op is a relational operator ($=$, $<$, $>$, $\leq$, $\geq$, $\neq$, ...), or an ensemblist operator ($\in$, $\notin$, ...) ; set is a set of values and val is a given value.

Example 6a. $RT_1 : Agency\_ID \in$ {'01903','01905','02256'}
$RT_2 : Year\_ID < 2001$
$RT_3 : Gender =$ 'F'

An aggregation rule is an "*if-then*" rule. The conclusion of the rule ("then" clause) defines the value of the generated attribute. The premise of the rule ("if" clause) is based on a composition of conjonctions or disjonctions of these rule terms:

$$r_{ij} \; : \; if \; RT_1 \; (and|or) \; RT_2 \; ... \; (and|or) \; RT_n \; then \; L_i^{sk}.A = val$$

Example 6b. The following rules define the values of the attribute `dim_type_agency` which characterizes the granularity level `AGENCY_TYPE`:

$(R1)$ $if \; Agency\_ID \in$ {'01903','01905','02256'} $then \; dim\_agency\_type =$ 'student'
$(R2)$ $if \; Agency\_ID \notin$ {'01903','01905','02256'} $then \; dim\_agency\_type =$ 'classical'

Example 6c. The following rule defines the value `'married women'` of the attribute `dim_persons_group` according to attributes `Marital_Status` and `Gender` of `CUSTOMER` table:

$if \; Marital\_Status =$ 'Married' $and \; Gender =$ 'F' $then \; dim\_persons\_group =$ 'married women'

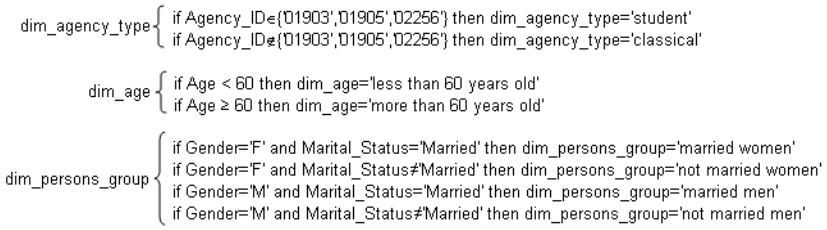

Figure 4. Evolving part of the *R-DW* data warehouse for the NBI analysis.

Thus aggregation rules create or enrich a dimension hierarchy by defining the values of the generated attribute according to a condition (Example 6b.) or a composition of conditions (Example 6c.). These rules translate the knowledge provided by users.

4. Case study

To validate our approach, we implemented a prototype of our *R-DW* model taking into account knowledge provided by a user. We developed a Web platform (HTML/PHP) behind Oracle DBMS used to store the fact table and dimension tables. One additional table contains the aggregation rules. The Web platform allows the user to define and visualize the rules that generate the analysis axes. It also allows the visualization of the query results. We applied our model on banking data for the NBI analysis. The fixed part of the model is represented in the Figure 1. Different dimension hierarchies are represented by the evolving part of the *R-DW* model in the Figure 4. Thus, the user will be able to run NBI analyses, not only by considering dimensions of the first level, but also by considering new granularity levels like agency type, age class...

In our first prototype, we have implemented the computation of aggregates for the analysis on the fly. We initially restricted ourselves to a decisionnal query with an aggregation according to one granularity level of a dimension hierarchy. This aggregation takes the form of a PL/SQL stored procedure of the DBMS. This procedure builds an aggregates table, by creating the SQL queries needed, according to the algorithm of the Figure 5. This algorithm can be used to answer the query “What is the NBI average for the different types of agency?”. The agency type is determined by two rules. Thus, the aggregates table contains two tuples corresponding to the NBI average for students and classical agencies. These values are obtained with the following SQL queries:

NBI average for student agencies:	*NBI average for classical agencies:*
SELECT MOY(NBI) FROM TF_NBI, AGENCY WHERE TF_NBI.Agency_ID=AGENCY.Agency_ID AND Agency_ID IN (‘01903’,‘01905’,‘02256’);	SELECT MOY(NBI) FROM TF_NBI, AGENCY WHERE TF_NBI.Agency_ID=AGENCY.Agency_ID AND Agency_ID NOT IN (‘01903’,‘01905’,‘02256’);

To obtain NBI average for student (resp. classical) agencies, in the WHERE clause of the query is the premise of the rule which defines a student (resp. classical) agency.

5. Related Work

A schema evolution is needed to take into account new analysis needs. According to the literature, schema evolution can be performed following two different ways, namely

```
Algorithm
Notation: The premise of the rule is written body(r_ij), and its conclu-
sion is written head(r_ij).
Input:  fact table F, set of aggregation rules R, attribute A, mea-
sure F.M_q, aggregation operator op,
Output: aggregates table TAgreg
Begin
    For each r_ij ∈ R
       If A ∈ head(r_ij) Then
          TAgreg =' SELECT op(F.M_q) FROM F WHERE body(r_ij)'
       End if
    End for
End
```

Figure 5. Aggregates computation algorithm.

schema updating and schema versioning. Schema updating consists in transforming data from an old schema into a new one [2,8]. In this case, only one schema is supported. On the contrary, schema versioning consists in keeping track of all versions of a schema [3,10]. In [1], the authors use versions too. They distinguish real version from alternative version. Alternative versions are used for simulating scenarios. They are derived by the data warehouse administrator from a previous version. Our *R-DW* model allows simulations too, but driven by users. Schema updating and versioning constitute a solution to the dimensions evolution, when the latter is induced by the evolution of the data themselves. However, once the data warehouse created, the users can only carry out analysis provided by the model. Thus they do not provide a solution to take into account new analysis needs which are driven by the expression of users' knowledge.

In spite of their disadvantages, these two approaches bring a certain temporal flexibility to the data warehouse model. Other works aimed at bringing flexibility within the data warehouse use mostly rule-based languages. Firstly, concerning data warehouse schema definition flexibility, two approaches are possible: using rules either to express the analysis needs [9] or the knowledge about data warehouse construction [11]. These different works propose to automatically generate the data warehouse schema from source schemas using rules but the schema's evolution is not taken into account. Create a new data warehouse when an analysis need appears is not a solution, even if this creation is automated. Secondly, the administrator must define some integrity constraints to ensure the consistency of not only the data, but also the analysis. In [5,7], the expressed integrity constraints use the data semantic to manage data inconsistency within the analysis. However the data semantic is not exploited for the analysis. The proposed approaches allow consistent analyses but their evolution is not evoked. Thirdly, in order to make the analysis more flexible, a rule-based language has been developed in [6] to manage exceptions during the aggregation process. This language allows to intentionally express redefinitions of aggregation hierarchies. We think that it is possible to treat these exceptions with the *R-DW* model.

Rule-based languages allow flexibility within data warehouses in different ways. It is precisely this flexibility which we seek for the analysis evolution. This evolution is conditioned by that of dimensions. The data warehouse schema updating or versioning constitute answers to the problem of dimensions evolution, when this latter is oriented by the evolution of data themselves. Moreover, the intervention of the administrator is

needed to implement these solutions. With the *R-DW* model, we bring a solution to take into account the emergence of new analysis needs directed by the expression of users' knowledge, without the administrator intervention, thus in a concurrent way.

6. Conclusion

In this paper, we proposed a new data warehouse model based on rules (*R-DW*), where rules allow us to introduce users' knowledge for analysis purposes. Our *R-DW* model is composed of two parts: one fixed part, which contains a fact table and first level dimensions and one evolving part, defined with rules, which generate granularity levels of dimension hierarchies. Thus the dimension hierarchies are generated according to users's knowledge therefore satisfying their analytical needs. The *R-DW* model presents the advantage of being able to dynamically create dimension hierarchies, without the administrator's intervention, in a concurrent way. We proposed a prototype of our *R-DW* model and applied it on the LCL banking data case study.

The perspectives opened by this study are numerous. First, we have to extend the analysis possibilities of our implementation. Then we intend to measure the performance of our approach in terms of storage space and response time. Furthermore we plan to define constraints on rules and a language that allows us to validate these rules. Moreover, in this paper, we presented an approach that involves the users in the analysis process by allowing them to integrate their knowledge in the system. We think it would be interesting to discovery new rules using non supervised learning methods.

References

[1] Bartosz Bebel, Johann Eder, Christian Koncilia, Tadeusz Morzy, and Robert Wrembel. Creation and management of versions in multiversion data warehouse. In *SAC*, pages 717–723, 2004.

[2] M. Blaschka, C. Sapia, and G. Höfling. On Schema Evolution in Multidimensional Databases. In *DaWaK'99: 1st International Conference on Data Warehousing and Knowledge Discovery*, pages 153–164, 1999.

[3] M. Body, M. Miquel, Y. Bédard, and A. Tchounikine. A Multidimensional and Multiversion Structure for OLAP Applications. In *DOLAP'02: 5th ACM International Workshop on Data Warehousing and OLAP*, 2002.

[4] D. R. Brown, Mark R. Cutkosky, and Jay M. Tenenbaum. Next-cut: A second generation framework for concurrent engineering. In *MIT-JSME Workshop*, pages 8–25, 1989.

[5] F. Carpani and R. Ruggia. An Integrity Constraints Language for a Conceptual Multidimensional Data Model. In *SEKE'01: XIII International Conference on Software Engineering Knowledge Engineering*, 2001.

[6] M. M. Espil and A. A. Vaisman. Efficient Intensional Redefinition of Aggregation Hierarchies in Multidimensional Databases. In *DOLAP'01: 4th ACM International Workshop on Data Warehousing and OLAP*, 2001.

[7] F. Ghozzi, F. Ravat, O. Teste, and G. Zurfluh. Constraints and Multidimensional Databases. In *ICEIS'03: 5th International Conference on Enterprise Information Systems*, pages 104–111, 2003.

[8] C. A. Hurtado, A. O. Mendelzon, and A. A. Vaisman. Updating OLAP Dimensions. In *DOLAP'99: 2nd ACM International Workshop on Data Warehousing and OLAP*, pages 60–66, 1999.

[9] H. J. Kim, T. H. Lee, S. G. Lee, and J. Chun. Automated Data Warehousing for Rule-Based CRM Systems. In *14th Australasian Database Conference on Database Technologies*, pages 67–73, 2003.

[10] Tadeusz Morzy and Robert Wrembel. On querying versions of multiversion data warehouse. In *DOLAP'04: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, pages 92–101, New York, NY, USA, 2004. ACM Press.

[11] V. Peralta, A. Illarze, and R. Ruggia. On the Applicability of Rules to Automate Data Warehouse Logical Design. In *CAiSE Workshops*, 2003.