ETL evolution from data sources to data warehouse using mediator data storage

Alexander Dolnik (alexander.dolnik@gmail.com)*

Saint-Petersburg State University

Abstract. The problem of evolution in the ETL process is investigated. There is no good solution in spite of numerous attempts to solve this problem. A new model to provide data warehouse and ETL scenario evolution is suggested. This model unite two different models of evolution to provide more flexible solution.

Key words: Design, model, ETL, data warehouse, evolution

1 Introduction

Rarely do data come to us in a form that is suitable for analysis and modeling. Sometimes data schemas are changed at the run time that leads to reconstruction an ETL process. Problem of ETL process evolution stays unsolved. The idea of this research is to unite two worlds of models: the model of the multidimentional databases evolution and the model of the ETL process.

2 Related Work

There are some methods for modelling of an ETL process and a data warehouse (for example [1]). In the work [1] authors suggest an extension of traditional UML for data description and ETL process design. Moreover, OMG defines MOF-based standard for modelling warehouses (CWM, [2]). The EER or ontology-based languages can be used instead of OMG modelling methods. For instance, approach suggested in the work [3] divides two levels: the conceptual level and the logical level. The conceptual level can be extracted from the ontology of an application (specification *SemanticWeb*). It is used for describing extern data sources and data warehouses. The logical level is used for describing ETL scenario logical operations . The logical level is defined using declarative language LDL++ ([4]). The big advantage of LDL++ language is its expressiveness.

An other method is suggested in paper [5]. The method is based on ETL process graph representation. Each graph node corresponds to a transformation model element and is annotated with What-If policies, containing instructions for changing graph structure. However, this approach is difficult to implement,

* This work was partially supported by RFBR (grant 07-07-00268a).

because of enormous amount of additional information required in nontrivial cases.

It is possible to describe evolution methods based on axioms [6], [7]. Actually, axiomatic approach can be used to formalize the part of the problem under consideration, but cannot solve it on a whole.

Entities change over time. Status of customers, classification rules, product characteristics and etc. lead to changes in the attributes of dimensions. In a transaction system, many times the change is overwritten and track of change is lost. But in data warehouse historical data are analizing. This problem has a very long history and a big merit of solving it has been made by Kimball ([8]).

3 Motivation example

According to the paper [9] the following example is mentioned. Patients' diagnoses for medical records are composed using "International Statistical Classification of Diseases and Related Health Problems" (ICD) code. The example of changing codes for diagnoses from ICD Version 9 to ICD Version 10 is considered. For instance, the code for "malignant neoplasm of stomach" has changed from 151 in ICD-9 to C16 in ICD-10. Other diagnoses were regrouped, e.g. "transient cerebral ischemic attacks" has moved from "Diseases of the circulatory system" to "Diseases of the nervous system". The same code described different diagnoses in ICD-9 and ICD-10. The granularity of codes changed.

The authors of paper [9] tried to answer on the following question: how can we obtain correct results for the queries like "did liver cancer increase over the last 5 years?" Of course, we should know about these changes to produce correct results but this condition is necessary. We extend this example to demonstrate another point of view – observe an evolution through the ETL process. Data sources cannot change immediately. Moreover, medical clinic does not give personalized and non-aggregated data. For instance, data from different clinics may be as it demonstrated below:

id	Medical clinic	Number of incidents	ICD code	ICD ver.	Date
1	clinic A	5	151	9	07/2009
2	clinic B	6	151	9	07/2009
3	clinic C	2	151	9	07/2009
4	**clinic A**	**2**	**151**	**9**	**15/07/2009**
5	**clinic C**	**2**	**151**	**9**	**15/07/2009**
6	clinic A	1	C16	10	08/2009
7	clinic B	5	151	9	08/2009

"Number of incidents" field comprises number of detected diseases from last mentioned period. For instance, in clinic B three cases of diseases are detected, they are denoted by ICD code 151 from 01/07/2009 to 01/08/2009. All values of field 'Date' except '15/07/2009' are generated inside ETL scenario by calling SysDate() function. So clinic A in comparison with clinic B changed ICD code

version after 15/07/2009. Clinic C is closed since 15/07/2009. Suppose that for analytic we need values aggregated by month. So such simple changes lead to the several reconstructions on the DW schema like: introduction a new classification of ICD code, changes in the member hierarchy, deletion of clinic. In addition, we lose an information about variety of the origin field 'Date' because some data are generated inside a process. That leads to some problems with versioning of data warehouse. The first problem is effectiveness. For example if we support two ICD versions we should convert data from version 9 to 10 (or contrariwise) on every query execution. The second problem is data sources availability limitations. For example clinic C does not send their data for 08/2009 but we want to build load curve for each clinic per month over year. The third problem is redundancy of data. We do not need to store abundance data in a data warehouse. For example for case of analytic function mentioned before we need data aggregated by month. Clinic C does not provide data for period 08/2009 but we have statistics up to 15/07/2009. We cannot operate with such data because we have not got appropriate transformation functions or enough data.

Sometimes it is better for administrator to delay evolution changes in data warehouse (may be with a losses of data). If it is possible new records are inserted into corresponding mediator data storage tables. Along with standart transformations of the ETL scenario transformation data from one mediator version to other may be presented or deducted. These functions can be transported to the data warehouse model as mapping functions. The functions are the part of the conceptual ETL process model as transformations.

4 Temporal multidimensional model for data warehouse

As it was mentioned above there are a lot of DW models that support data evolution over time. Now we concentrate on two works: [9] and [10]. The idea of these works is quite simple. First of all, we should realize versioning to execute (if it is possible) 'old'/'new' analytical queries on the 'old' data as the 'new' data. That means unity use of data that have different versions. But the real data should base on a special temporally consistent fact tables. In such tables facts are stored. Different facts may be non-valid or have different meaning in different time slices. Using special 'extension' functions we can 'extend' lifetime of these facts to another time slices. To construct a multidimentional cube we need dimentions that can evolve over time. Let us suppose that we have n temporal dimensions $D_1, D_2, \ldots, D_n$, a time dimension T. The dimention may has a hierarchical structure and be represented as a directed acyclic graph (DAG). Such DAG is also known as categories graph. Each category consists of a set of the other categories or contains a set of measures. So dimension may be depicted as a set of m measures $M = \{m_1, m_2, \ldots, m_m\}$ that represented all leafs in the DAG. In accordance with the paper [9] user defined attributes are also permitted for dimensions but for our task it is not essential.

So a temporally consistent fact table f can be seen as a function:

$$f : D_1 \times D_2 \times \ldots \times D_n \times T \rightarrow dom(m_1) \times dom(m_2) \times \ldots \times dom(m_m)$$

where D_i is the set of members of each dimension D_i and $dom(m_k)$ is the range for the measure m_k. We, likewise in paper [10], introduce a confidence factor. The confidence factor is a value that describes the reliability of data. It plays a crucial role in evaluating state of mapped data in qualitative way.

The mapping relationship binds two contiguous versions of a value. Mapping can be from one member to other member or category. Also in mapping should be declared transformation function. This function specify how the value must be mapped. Moreover, the confidence factor for this function should be introduced. Confidence factors are linked to mapping functions. To support mapping between categories mappings are calculated from the aggregation function of their children values. In the dimention hierarchy DAG some nodes may have numeric parameter level. By default for each node level number is equal to depth in the DAG. Level is used to provide correctness for aggregation functions. A query execution requires some mode of presentation. This mode imposes constraints on using versions. To be sure in correctness query execution the authors of paper [9] introduce integrity constraints. Summarize integrity rules we can emphasize the following features: a temporal granularity of the model and an argument uniqueness law. The temporal granularity means that the value can be send to the lower level when the valid time intervals of two levels are intersect. The argument uniqueness law means that we must deny such situations when at the query execution the value has been considered more than one. The aggregation function is a traditional operation with the data. The theory of aggregation functions is well developed ([11]) with the exception of holistic aggregation functions type. To provide correctness we must provide aggregation functions with a confidence factor.

In addition, four features should be satisfied:

- Temporal Data Warehousing. We have discussed about this above.
- Fact-Constellation Schemas. This property means permission for construction galaxy schemas (with several fact tables) in spite of using pure star or snowflake schemas.
- Proportional Aggregation. A good formalization can be found at the paper [11]. As an example, the hierarchy of a day may has non homogeneous granularity in different schemas. In one schema the day is divided into a non-working day and working one. Working day, in its turn, consist of the morning and the evening. In other schema the day is divided into the day inside the term and the day outside the term. The day inside the term consists of the morning and the evening.
- Generic Dimensionality. E.F. Codd (at the paper [12]) formulated sixth OLAP rule of "Generic Dimensionality" as each dimension must be equivalent in both its structure and operational capabilities.

To modify the structure of a temporal multidimensional schema, authors of the paper [10] provide four basic operators on DW schema: insert, exclude, associate and reclassify. According to the paper [10] the administrator integrates changes in the structure by means of these operators. In the following sections we show how to integrate schema changes with the ETL scenario.

5 ETL process evolution model

When we consider the temporal multidimensional model for data warehouse separately from the ETL process we deal with a semantic gap. The semantic gap between ETL process and DW is obvious because changes that occur into the data sources should drill through ETL process to achieve DW. Some necessary information may be lost. That is why we need a new ETL process model. In the paper [3] ETL process development is divided into two desing steps: conceptual and logical modeling of ETL workflows. At the first stage they construct conceptual level using graph-based representation, called datastore graph. This technique is a generic data model allowing the representation of the several types of schemas. A graph representation, termed ontology graph, is introduced for the application ontology. An annotation of a datastore is accomplished by formally defining mappings between the nodes of the datastore graph and the ontology graph. These mappings can be represented as labels assigned to the nodes of the data store graph. Some additional information may be included into ontology on this stage, like: name of data source, number of datasource records, data field type, max/min values of the field, data patterns and etc. We divided each datastore graph into three levels:

- templates conceptual schema that represents a set of input data sources schemas;
- data warehouse conceptual schema;
- common area conceptual model that represents an ETL process model (mapping templates into data warehouse).

The datastore graph levels are depicted on figure 1.

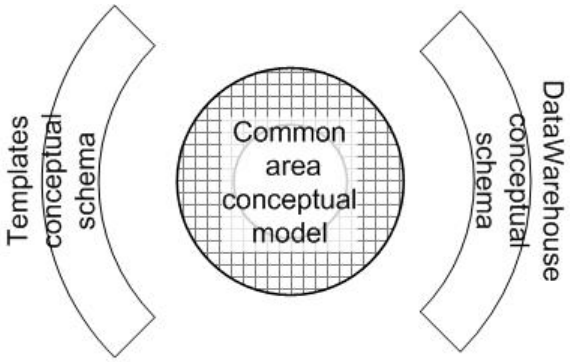

Fig. 1. Schema of Common Conceptual Model

We use the temporal multidimensional schema to represent conceptual schema for data warehouse as it was described in section 4. But the conceptual schema

for data warehouse is a part of an ontology. A short definition of the ontology is a specification of a conceptualization. An ontology assignment identifies ontology itself. Our ontology is for data mapping deduction, reasoning techniques of conditions and constraints. It also helps to control data evolution. So it is ontology for data schema manipulation.

The third component of the datastore graph is a common area conceptual model. This is the middle level that helps to transfer data from data sources to the DW. It plays a mediator role. To maintain our ETL process temporary data the storage is created. In spite of repository for metadata and process we use the temporary data storage to maintain ancillary data.

As early as 1996, Kimball introduce three types of "Slowly Changing Dimensions" [8] (also known as SCDs). Dealing with these issues involves SCD management methodologies referred to as Type 0, 1, 2, 3, 4, and 6. Type 6 SCDs are also sometimes called Hybrid SCDs. The most common slowly changing dimensions are Types 1, 2, and 3. The first slowly changing dimension method is to override the old value. For instance if an insurance policy status has moved from 'Lapsed' to 'Re-instated' the new status is over written on the old status. This is obviously done, when we are not analyzing the historical information. As it has been said earlier, this method is used in transaction systems (or data sources). So input data evolve using slowly changing dimension type one. The second slowly changing dimension method is to add new record with a separate identifier as the primary key. In other words we keep track of a data. The third slowly changing dimension method is to add a new field 'old attribute value' (but not a new record). However, this has limitations. This method has to know from the beginning on what attributes will change. As we see at the section 4 this problem has been solved by introducing mapping and valid time.

So to propogate evolution changes from data sources to data warehouse we need additional data storage that can support slowly changing dimension type three. It stores some data from previous execution of the ETL process. We call this datastore as a mediator data storage. The main functions of the mediator data storage is to provide correctness of the DW model.

The main requirements for mediator are:

- All data from data sources should pass through mediator datastore
- Mediator datastore should only contain such information that may be helpful to support DW version model evolution
- Minimization of dubbing data

Conceptual model is not enough to construct ETL process. To provide transformations we need a logical model. In industry these transformations have been made by different scripting languages. And often such transformation looks like black-box with unknown semantic. Sometimes these black-box can produce 'new fields'. For example, in our motivation example we use SysDate() function to produce timestamp of the execution ETL scenario date. Without knowing about sematic of the black-box function it is hard to make the modifications. That is why we should carefully choose the data transformation language. From our

point of view such language as LDL++ ([4]) is convenient. Of course, we cannot exclude all black-box functions but we should minimize their number.

6 Mediator conceptual data storage

Mediator data storage helps to make data warehousing versioning easier and effectiveness. That may be achieved by storing some data inside ETL process for a while.

Let us suppose that we already have mapping between data sources and the data warehouse (section 5). Of course administrator of the data warehouse should point significant for analitical functions fields and notions. That should be done on the conceptual level. Then we lock at the datastore graph and construct mediator schema.

A data mediator schema at the initial stage is constructed using rules:

- nodes of the datastore graph should be on a way from data sources to data warehouse;
- each node of the mediator schema must obtain data only from one data source (for each node only one data source ancestor);
- select only such nodes on a way that have the maximal depth from data sources value nodes (deepest nodes).

Then mediator schema may evaluate. All data are come through mediator schema. Some of them may be stored there.

We suggest the following criteria to detect which data should be saved into the mediator data storage:

- Evaluation of affected data in DW (for simple case it is number of records);
- Maximal length of stable interval for all analysis functions;
- Number of mapping functions in the DW schema.

As for our example the stable interval does not increase if data from bold fields are added. In addition number of mapping functions in the DW grows up if we want to support two versions of ICD code. That is why we add these data into mediator data storage.

In mediator data storage we use Kimball's ([8]) slowly changing dimensions type two with the following modification: from time to time we cleanup unnecessary data. The algorithm of detecting these data will be described below.

Let us introduce some auxiliaries structures for mediator data storage. These structures help to track history of evolution classification in data sources. The first case is when data source use extern classification it must contain a classification version. Usually, this classification schema is stored in repository. So there is no problem to detect changes in classification in this case. The second case is when data source contains implicit classification and integrity constraints. Note that these integrity constraints are valid inside the ODS (operational data source) but may be break outside and in time. For instance, an enterprise contains classification for its divisions. Every period under report enterprise give

statistic for each division. So it has an integrity constraint that will be detected at the initial stage describing in the section 5. If we trust the data we can observe evolution process in this enterprise. To bring such cases into our model we create special table in the mediator data storage. This table contains fields: new surrogate key, validity period, initial composite key (that consist of enterprise name and division name) and field that shows does constraint true or false for the current data.

When a new data are come into data staging area we use data in mediator storage to detect changes. First of all, we detect a subset of fields that are charged with structure of the multidimensional model for data warehouse. These fields form dimensions and hierarchy structure of the dimension (directed acyclic graph) as it was mentioned in the section 4. Then we compare the validity period for the selected fields records that have the same initial primary key or initial composite key. This way traces changes that occur with the source data which is not fixed in data warehouse.

After obtaining a changed records they may be propagated into data warehouse with the basic structure evolution operators described in the section 4. Moreover we can automaticly detected a type of evolution operations. The old data in mediator data storage are replaced by recent data at the time of propagation changes into data warehouse.

7 Conclusion and Future Work

We introduce a new conceptual model with temporary mediator data storage. The next step is to study criteria and mediator data storage properties. From our point of view, approach of a combination ETL process and DW models is very promising. The suggested model with mediator data storage can solve a lot of existing problems but practical validation of this model should be done.

References

1. Lujan-Mora, S., Vassiliadis, P., Trujillo, J.: Data mapping diagrams for data warehouse design with uml
2. group, O.: Common Warehouse Metamodel(CWM) Specification. (2001), ee = http://www.omg.org/spec/CWM/1.1/)
3. Sellis, T.K., Simitsis, A.: Etl workflows: From formal specification to optimization. In: ADBIS. (2007) 1–11
4. Zaniolo, C.: LDL ++ Tutorial. (1998), ee = http://www.cs.ucla.edu/ldl/tutorial/ldlcourse.html)
5. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: What-if analysis for data warehouse evolution. In: DaWaK. (2007) 23–33
6. Peters, R.J., Özsu, M.T.: An axiomatic model of dynamic schema evolution in objectbase systems. ACM Trans. Database Syst. **22**(1) (1997) 75–114
7. Simanovsky, A.: Evolution of schema of xml-documents stored in a relational database. In Barzdins, J., ed.: Proc. of the Baltic DB IS'2004. Volume 672., Riga, Latvia, Scientific Papers University of Latvia (June 2004) 192–204

8. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley (1996)
9. Eder, J., Koncilia, C., Morzy, T.: The comet metamodel for temporal data warehouses. In: CAiSE '02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering, London, UK, Springer-Verlag (2002) 83–99
10. Body, M., Miquel, M., Bedard, Y., Tchounikine, A.: Handling evolutions in multidimensional structures. Data Engineering, International Conference on **0** (2003) 581
11. Lenz, H.J., Thalheim, B.: Olap databases and aggregation functions. In: SSDBM '01: Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management, Washington, DC, USA, IEEE Computer Society (2001) 91
12. Codd E.F., C.S., C.T., S.: Providing olap (on-line analytical processing) to user-analysts: An it mandate. Codd & Date, Inc **31** (1993) 1–31