ABSTRACT: Warehousing projects can know cases of failure because they did not take into consideration the users’ needs especially if they are not experienced with the technologies of data warehouses. The solution consists on building the data warehouse incrementally by designing and implementing one data mart at a time. In this work we propose a method to design data mart schema from OLAP requirements that are presented as schemas and grouped according to their domain. We apply the data integration technique to merge the different schemas so that we get one data mart schema by group. The integration is composed by schema matching (detecting semantic correspondence and the conflicts) and schema mapping (solving the existing conflict).

Keywords: OLAP Requirements Schema, Data Mart design, Schema Integration, Schema Matching, Schema Mapping

Received: 2 February 2013, Revised 8 March 2013, Accepted 17 March 2013

© 2013 DLINE. All rights reserved

1. Introduction

Warehousing projects may know cases of failures during their achievements. As causes we can mention the users’ needs that are generally poorly expressed by either designers or developers, also, the users are, in many cases, not experienced with the technologies of DWs. The solution is building the data warehouse incrementally by designing and implementing one data mart at a time [1]. Data mart plays a crucial role when using the bottom-up approach to design the data warehouse. It helps to identify, validate and convince the final users of the potential benefits [14]. In order to increase the efficiency and the success of the data warehouse, it is necessary to have methodological framework for data mart design.

The data mart “is defined as a flexible set of data, ideally based on the most atomic (granular) data possible to extract from an operational source, and presented in a symmetric (dimensional) model that is most resilient when faced with unexpected user queries” [19]. It is accessed directly by end users, and its data is structured in a way that is easy for users to understand and use [6].

We propose in this work a new method for designing data mart that starts from OLAP Requirements (ORS) to take into consideration the users’ needs. To achieve our goal, we suppose that ORSs belong to the same domain to facilitate the generation of our schema since the data mart should be constructed for a specific business line or team. The ORSs are different in the structure and semantic, and to take this criterion into consideration, we use the schema integration technique, so that we can get one data mart schema from existing ones.
This technique is used to find all relationships between the different schemas which will be merged. The corresponding process is not an easy task and the basic problems are mainly because of the structural and semantic diversities of schemas that will be merged. Three problems related to the schema integration exist: data model heterogeneity (no guarantee that the data schemas share a common data model), structure heterogeneity (equivalent business concepts are modeled using different models in the same data model) and semantic heterogeneity (the difference related to the interpretation of real words concepts).

To overcome the first problem, we specify a common structure interface to the different users, so we are sure that the different schemas have the same data model, and the common structure schema model serves also to solve the second problem since it defines their categories i.e. the user defines if he needs a specific term as a fact, dimension, level, etc. We still have the semantic heterogeneity. This latter will be solved next using different techniques such the schema matching, mapping, etc.

There are two strategies behind the creation of a global schema using the schema integration which are “bottom-up” and “top-down”. The use of one of them depends on the existence or not of the global schema. So, in the first strategy, the global schema does not exist, and the integration process involves both the definition of a global schema, as well as the definition of the mappings between the data source schemas and the global schema [4]. In top-down integration setting the global schema exists, and mappings need to be defined between the data source schemas and this global schema [15]. The bottom up strategy is appropriate in the case of schema integration process while the top-down is more suited from the perspective of domain engineering [11]. So, and according to our goal we will adapt bottom up strategy to create our global schema. There are different ways to apply this strategy. In fact, it depends on how we will merge the local schemas i.e. using as input two schemas (binary) or all-at-once (n-ary). The binary can be divided into “ladder” [3] and “balanced” [12]. The n-ary is composed by “one-shot” [20] and “iterative” [21].

The outline of this work is as following:

- In section 2, we give a short description of some works presenting methodologies of data mart design.
- In section 3, we define the structure of the schema that we will use in the rest of this paper.
- In section 4, we specify the formula used to calculate the degree of similarity of the elements exiting in the schemas.
- In section 5, we present our solution that consists on merging the schemas using the schema integration technique that uses the schema matching to detect the semantic correspondence and the conflicts also the schema mapping technique to solve the existing conflicts to get a single schema at the end.
- Finally, we finish this work with a conclusion and perspectives.

2. The State of the ART

In the literature, different approaches have been proposed to ensure the design of the data mart. In the following we will present a short summary of some of those works.

In [13], the authors propose a method for generating the schemas staring from ER schemas. Their method is composed by 6 steps. It starts by extracting the fact that corresponds to an entity or an n-ary relationship. Then, it builds the attribute tree that corresponds to quasi-tree such that: each vertex corresponds to an attribute of the scheme, and the root corresponds to the identifier (primary key). It applies the pruning and grafting to illuminate the unnecessary attributes and levels of detail, where pruning drops any subtree from the quasi-tree, and grafting preserves the descendants. In the next step, it defines the dimensions that are chosen from the attribute tree among the children vertices of the root. Then it defines the measures by applying aggregation functions to numerical attributes of the attribute tree. Finally it defines hierarchies, and this is done through the arrangement of the attributes into a quasi-tree such that a -to-one relationships holds between each node and its descendants.

In [6], the authors propose a method to design data mart based on an enterprise data model represented in ER form. It starts by classifying the entities into three categories: transaction entities, Component entities and Classification entities. In the second step, it defines the hierarchies that correspond to the set of tables related through one-to-many relationships and aligned in the same direction. The third step produces dimensional models using operators (collapse hierarchy, aggregation) or design model option.

The fourth step evaluates and refines the schema, it is an iterative process. It consists at: combining fact tables, combining dimensional tables, manage the Many-to-Many Relationships (as solutions: ignore the intersection entity, convert the many-to-
many relationship to a one-to-many relationship, etc), and handling Subtypes.

In [10], the authors propose a method for developing data marts from existing ER schemes. The proposed method is composed by 4 steps: Identification of facts and dimensions: it extracts the facts (entities, relationships or attributes), dimensions (are a sub-scheme of the given ER scheme) and measures (are atomic properties of a fact). The restructuring of ER scheme: it reorganizes the original ER where the facts are represented as entities and the dimensions are added. Within each dimension, the various level of aggregation is represented in an explicit way. This step requires performing one of the following transformations: replacing many-to-many relationships, adding new concepts, selecting a simple identifier for each level entity and removing irrelevant concepts. The derivation of dimensional graph: the dimension graph represents facts and dimensions of the restructured ER scheme. If the node is an entity, it represents the domain of the key of the entity, and if it is an attribute, it represents the domain of the attribute. The different nodes are linked through an arc that corresponds to a specific function. Finally the translation into MD model: the MD can be directly derived from dimensional graph. There is an MD dimension for each dimensions of the dimensional graph. For each dimension, there is a MD level for each node. For each arc there is a roll-up function. The number of atomic dimensions represents measure nodes and descriptive nodes.

In [7], the authors propose a hybrid approach to build the data mart schema. The decision-makers needs are expressed from data sources, and the data mart schema is elaborating by confronting decision-making needs with data sources. The proposed process is composed by four successive steps (each one produces a new schema more complete than the previous one): First step specifies the facts: it extracts from the source CD a limited set of candidate facts. The user (decision-maker) chooses one of the proposed ones and he specified the required aggregation functions. The second step specifies the dimensions: it proposed all possible dimensions associated with the chosen fact. The user indicates the dimensions that he needs. Third step specifies the hierarchies: it proposes all possible hierarchies for each dimension. The user chooses the appropriate hierarchy. The last step elaborates the data mart schema and saves the multidimensional elements into meta-data. This latter will be used to choose the candidates facts from the source schema in the first step.

In [2] the authors present a method Mixed to support the identification and design of data marts. The proposed method is composed by three basic steps: the Top-Down analysis emphasizes the user requirements. The user requirements are collected through interviews that help to gather information from business analysts and/or managers about the company goals and needs. The Bottom-down analysis brings to surface the semantics of the existing operational databases. This step is devoted to examining the conceptual model of the operational databases in order to extract the candidate star schemas for the data warehouse. This approach is based on an exhaustive graph analysis technique that exploits an ER representation of the conceptual schema of the operational databases. The candidate star schemas are structured as an aggregation of data around “additive” data items. This step can generate a large number of candidates. And finally the integration integrates the two viewpoints and generates feasible solution. It matches each ideal schema determined by the first step with all the candidate star schemas produced by the second step, and ranks them according to a metrics. The designer’s responsibility to choose the candidates that fit the ideal schema.

3. The Schema Structure

Since a schema is complex in term of composition, comparing its whole structure at once is not an adequate solution, so we propose its decomposition into set of categories which are: fact, dimension, measure, attribute, parameter, and hierarchy.

- The Fact corresponds to the subject of analysis. It is defined by a tuple \((FN, M^F)\) with \(FN\) represents the name of the fact and \(M^F\{m_1, m_2, m_3, m_4,\ldots\}\) corresponds to the set of measures related to the fact \(F\).

- The Dimension represents the axis of analysis. It is composed by \((DN, A\{\}, H^D)\) with \(DN\) corresponds to the dimension name, \(A\{a_1, a_2, a_3, a_4,\ldots\}\) presents the set of attributes describing the current dimension \(D\), and \(H^D\{h_1^D, h_2^D, h_3^D, h_4^D, \ldots\}\) is a set of ordered hierarchies. Each hierarchy has a tuple \((HN, P\{\})\) with \(HN\) is the name of the current hierarchy and \(P\{p_1, p_2, p_3, p_4,\ldots\}\) is a set of ordered parameters.

4. Degree of Similarity (DESIM)

In this section, we present the formula used to define the “Degree of Similarity (DeSim)”. 

80 International Journal of Computational Linguistics Research Volume 4 Number 2 June 2013
When we calculate the similarity of two schemas, we should take into consideration the following points:

- **The identical**: the case where we use the same elements name in the two schemas.
  
  \[ \text{DeId} (e_1, e_2) = 1 \text{ if } e_1 \text{ and } e_2 \text{ are identical and 0 else.} \]

- **The synonymous**: it is the case where we use two different names that have the same meaning.
  
  \[ \text{DeSy} (e_1, e_2) = 1 \text{ if } e_1 \text{ and } e_2 \text{ are synonymous, and 0 else.} \]

- **The typos**: it is the case where the user makes mistakes when writing the name of the element.
  
  In this case, we calculate the degree of error. If it is low, we are in the case of typing error. If it is high we are in the case of two different words. In the following we only take into consideration the first case.
  
  \[ \text{DeTy} (e_1, e_2) = 1 \text{ if } e_1 \text{ and } e_2 \text{ are the same with the existence of typing error.} \]

- **The post-and pre-fixe**: it is the case where we use post-fixes or pre-fixes to design the same thing.
  
  \[ \text{DePost} (e_1, e_2) = 1 \text{ if one the two elements is the post-fixe of the other, and 0 else.} \]
  
  \[ \text{DePre} (e_1, e_2) = 1 \text{ if one of the elements is the pre-fixe of the other, and 0 else.} \]

Let \( \text{Sch}_1 \) and \( \text{Sch}_2 \) be two schemas belonging to the same domain.

Let \( C_i \) be the categories of elements existing in the schema. \( C_i \) can be: fact, dimension, measure, attribute, parameter, and hierarchy: \( \forall e_i \in \text{Sch}_1, \exists e_j \in \text{Sch}_2, \) such that \( e_i \) and \( e_j \) belong to the same category \( C_i \).

The degree of similarity between \( e_i \) and \( e_j \) \( (\text{DeSim} (e_i, e_j)) \) measured by the numeric value in \([0, 1]\):

\[ \text{DeSim} (e_i, e_j) : \text{Sch}_1 \times \text{Sch}_2 \rightarrow [0, 1] \]

The degree of similarity is the average of previous degrees, and it is calculated as follow:

\[ \text{DeSim} (e_1, e_2) = [\text{DeId} (e_1, e_2) + \text{DeSy} (e_1, e_2) + \text{SeTy} (e_1, e_2) + [\text{DePost} (e_1, e_2) \text{ or } \text{DePre} (e_1, e_2)]] / 4 \]

Two schemas are considered “Similar” if they have the highest degree of similarity.

Two similar schemas can be totally or partially merged.

- **Total Merge (TM)** implies the two schemas have the same elements. \( \text{TM} (\text{sch}_1, \text{sch}_2) = \{ \forall x_1 \in \text{sch}_1, \exists x_2 \in \text{sch}_2 / x_1 \equiv x_2 \} \)

- **Partial Merge (PM)** implies the two schemas have some elements in common. \( \text{PM} (\text{sch}_1, \text{sch}_2) = \{ \exists x_1 \in \text{sch}_1, \exists x_2 \in \text{sch}_2 / x_1 \equiv x_2 \} \).

5. Schema Integration Phase

In this section we will present our methodology that ensures the construction of the global schema by merging the set of local schemas belonging to the same domain.

The proposed methodology is composed by two steps: the first one consists in comparing the schemas to determine the elements that are semantically related. It serves also to detect the conflicts (if they exist). In the second step, we start by solving the conflicts, and then we merge the schemas.

5.1 Schema Comparison

In the schema comparison step, we propose the use of the schema matching technique to detect the semantic correspondence, as well as the conflicts that may exist in such case.

5.1.1 Schema Matching

The schema matching is considered as one of the basic operations required by the process of data integration [18]. It is used to solve the problem related to the heterogeneity of the data sources by finding semantic correspondence between the elements.
of the two schemas. This phase takes as input two schemas to get as output set of elements that will be mapped. In the literature, it is considered as challenging task for the following reasons [9]:

- Different schemas presenting identical concepts can have different structure also different names
- They can contain similar but non-identical concepts
- They can be expressed using different models.
- They can use similar words to have different meanings.
- Etc.

To ensure the effective schema matching tool, we should take into consideration the combination of several techniques such as: linguistic matching of names of schemas elements, the comparison of the instance of data having similar structure. In this level of our work, we need to focus on the first technique, and according to [23], it proceeds in three steps: normalization, categorization and comparison.

- **Normalization**: the difference of names can be because of the use of abbreviations, acronyms, punctuation, etc. They perform tokenization (i.e. parsing names into tokens based on punctuation, case, etc), expansion (identification of the abbreviation, acronyms, etc). So to take the previous steps into consideration we propose the use of domain ontology, levenshtein name, etc.

- **Categorization**: the elements composing the schemas are clusters into categories. In our case we have the following categories: fact, dimension, measures, attributes, hierarchies, parameters. Each element of the schema belongs to a specific category.

- **Comparison**: a coefficient of linguistic similarity is calculated by comparing the tokens extracted from the names of the elements.

Clustering the elements into categories reduces the number of one-to-one comparison eliminating the unnecessary comparisons (for example: comparing a fact element with a dimension element).

At the end of this phase we will get a table containing set of coefficients calculating the similarity between the elements belonging to the same category. The coefficient is in the range \([0, 1]\) with “1” implies a perfect linguistic match.

To compare the schema, we propose the division of the categories of the schemas into two types: the first one includes fact, dimensions, measures and levels and the second type includes the hierarchy. The identification is done, then, object by object in function of its category (fact of sch1 against fact of sch2, dimension of sch1 against dimension of sch2, etc) except for the hierarchy where we have to take into consideration the relationships of the parameters of the hierarchies also their order.

### 5.1.2 Schema Matching Algorithm

This algorithm serves to extract the matching elements to facilitate their merging next. This is done by calculating the similarity degree between the elements as follow:

We start the comparison with the “fact” if the two facts of the two schemas are equivalent we move to the comparison of the measures. If they are equivalent, we are in the case where the two schemas deal with the same fact information. The resulting schema will be composed by one fact table and a set of measures of one of the two schemas. In the case where the measures are different, the fact table will contain the combination of all the existing measures. When the two facts are different, the resulting schema will contain the two fact tables.

Next, we move to compare the “dimensions”. We propose in this level the use of similarity matrix. The columns contain the names of the dimensions of the first schema and the lines contain the name of the dimensions of the second schema. The cells contain the “DeSim” that corresponds to the degree of similarity between the elements of two schemas.

When two dimensions of two different schemas are equivalent, we compare the “attributes”, then the “parameters” of the hierarchies. If they are equivalent we keep each one of them, else we combine them. In all of the previous comparison cases, we use similarity matrix (Figure 1) as a way to find the closest elements, and for the hierarchies, we should take into consideration the order of the elements.
5.1.3 The Conflicts Detection

The previous step helps to identify the elements that are semantically related but this is not sufficient to integrate the set of local schemas into global one. What we need now is extracting the schemas conflicts and dealing with them to satisfy the requirements. The author in [11] presents three types of conflicts that occur during the integration phase: extensional, structural and naming conflicts.

- **Extensional conflict:** it refers to the redundancies among different classes [22]. There are four types of extensional relationship. The authors in [8] present them and they give the solution for each type.

Let $E_A$ and $E_B$ two elements extracted from the two schemas and belonging to the same category, and $K_{EA}, K_{EB}$.

- **Equivalent sets:** $E_A \equiv E_B$ (there is no conflict). The two elements present the same instance e.g. “employee”, “worker”, “Staff_Member”.

- **Subset relationships:** $E_A \subset E_B$; $E_A$ represents a subset of $E_B$ instances at all times. e.g. “Employee”, “Manager”. The solution: $K_{EA}$ inherits from $K_{EB}$

- **Overlapping sets:** $E_A \subset E_B \cap \neq \phi$ and $E_A \cap \neq \phi$ and $E_B \cap \neq \phi$. A and B can share the same instances. e.g. “Employee” and “Client”. The solution: $E_A$ and $E_B$ inherit from the new class $K_{EA \cup EB}$.

- **Disjoint sets:** $E_A \cap E_B \neq \phi$ e.g. $E_A$ and $E_B$ represent a set of instances but they do not share it at any time. e.g. “EmployeeMS” and “EmployeeDO”. The solution $E_A$ and $E_B$ inherit from the new class $K_{EA \cup EB}$.

- **Structural conflict:** In the context of the schemas of databases, the structure conflict “occurs when related real world concepts are modeled using different constructs in the different schemas” [16].

The authors, in [16], extract from the literature the following types of structural conflicts that are specified to ER schema.

1. An entity type in one schema is modeled as an attribute of an entity type or a relationship set in another schema,
2. An entity type in one schema is modeled as a relationship set in another schema,
3. A relationship set in one schema is modeled as an attribute of an entity type or a relationship set in another schema,
4. An attribute of a relationship set is modeled as an attribute of an entity type.

In the context of our work, we do not need to focus on this kind of conflict since we will keep each element as it is in the global schema.

- **Naming conflict:** According to [5], it “refers to the relationship between the object attribute or instance names”. The relationship between the names is commutative i.e. term 1 is homonyms of term 2 implies also term 2 is homonyms of term 1.

In this part, we treat homonyms and synonyms. The homonyms occur if one name is used for two or more concepts [17], and the synonyms occur if two or more names are used for the same concept [17], it can exist in any category. It is solved using the generalization [5].

This conflict is determined using different tools such as wordnet, thesaurus, etc. Their specification depends on their context.
5.2 Schema Mapping

Once we detect the conflicts existing between the two schemas, we move to the next step that consists on resolving those conflicts using schema mapping technique. This latter is used to specify the relationships between two types of schemas: the source and the target.

In our case we have two sources (the inputs) “sch1” and “sch2”, and one target (the output) “T”. $M = (sch1; sch2; T; \delta)$

**Definition**: a schema mapping is a qua-triple $M = (sch1; sch2; T; \delta)$ such that “sch1” is the first schema, “sch2” is the second schema, “T” is the target schema (the schema resulting from the merging of the two input schemas), and $\delta$ is a set of formulas over <sch1, sch2; T>.

- An instance of $M$ is an instance of <s1, s2; t; $\delta_i$> over <sch1, sch2; T; $\delta_i$> that has a specific formula in the set $\delta_i$.
- Let Ins< M > denotes the instances <s1, s2; t; $\delta_i$> of $M$. Each instance has its own formula $\delta_i$.

The formulas existing in $\delta_i$ correspond to one of the following functions:

- **Union**: $R = \text{union} (e1, e2)$ implies that $R$ is the union of the two elements $e1$ and $e2$. This function can take as input more than two elements but since we propose the use of binary ladder, we need two elements as input. $R$ contains all the components of $e1$ and all components of $e2$.

- **Intersection**: $R = \text{intersection} (e1, e2)$ implies that $R$ is the intersection of the two elements $e1$ and $e2$. $R$ contains the components that exist in $e1$ and $e2$.

- **Disjoint**: $\text{disjoint} (e1, e2)$ $e1$ and $e2$ are disjoint if they no component in common.

The schema mapping is generally done manually and it requires good domain knowledge. Even the applications that have been developed to facilitate this task, they visualize the sources and it is the role of the user to finish this task.

We use the mapping as an intermediate step for merging the schemas sources. In the following, we present the set of rules that serves to map two schemas. Here the task of user consists on confirming the result or modifying it if it is necessary.

- If the facts are similar, then the user chooses one of the existing tables (intersection).
  - If the measures are similar, the user chooses (intersection).
  - If the measures are different, the fact table includes all the existing measures (union).

- If the facts are different, the system keeps the two tables separately. Each table contains its measures (disjoint).

- For each couple of dimensions:
  - If the dimensions are similar, the user chooses one of them (intersection)
    - If the attributes are similar, the user chooses his needs (intersection)
    - If the attribute are different, the system units the attributes (union).
    - If the hierarchies are similar, The user selects one of the two dimensions including its attributes and hierarchies
    - If the hierarchies are different, the system keeps the two hierarchies and links them to the same dimension
  - If the dimensions are different, the system keeps the two tables separated. Each one with its attributes and hierarchies

6. Conclusion

In this work we proposed a new method that ensures the design of data mart from OLAP requirements. By this way we can be sure that the result schema meets the user requirements.

Our method starts from OLAP requirements schemas; it merges them using the schema integration technique to get a one schema at the end that corresponds to the schema of the data mart. The used technique is divided into two parts: in the first one and using the matching techniques we extract the correspondence semantics and the conflicts, and in the second part we
resolve the conflicts and we merge the schema.

As perspectives, we propose the validation of this schema by its confrontation with the existing data sources.

7. Acknowledgment

I would like to thank everyone!

References


