ORE: An Iterative Approach to the Design and Evolution of Multi-Dimensional Schemas

Petar Jovanovic
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
petar@essi.upc.edu

Oscar Romero
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
oromero@essi.upc.edu

Alkis Simitsis
HP Labs
Palo Alto, CA, USA
alkis@hp.com

Alberto Abelló
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
aabello@essi.upc.edu

ABSTRACT

Designing a data warehouse (DW) highly depends on the information requirements of its business users. However, tailoring a DW design that satisfies all business requirements is not an easy task. In addition, complex and evolving business environments result in a continuous emergence of new or changed business needs. Furthermore, for building a correct multidimensional (MD) schema for a DW, the designer should deal with the semantics and heterogeneity of the underlying data sources. To cope with such an inevitable complexity, both at the beginning of the design process and when a potential evolution event occurs, in this paper we present a semi-automatic method, named *ORE*, for constructing the MD schema in an iterative fashion based on the information requirements. In our approach, we consider each requirement separately and incrementally build the unified MD schema satisfying the entire set of requirements.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Data warehouse and repository*

Keywords

Data Warehouse, Multi-Dimensional Design, ETL Design

1. INTRODUCTION

Data warehousing ecosystems have been widely recognized to successfully support strategic decision making in complex business environments. One of their most important goals is to capture the relevant organization data provided through different sources and in various formats and to enable analytical processing of this data. The most common design approach suggests building a centralized decision support repository (like a DW) which gathers the organization's data and which, due to its analytical tasks, follows a multidimensional (MD) design. The MD design is distinguished by the fact/dimension dichotomy, where facts represent the subjects of analysis and dimensions show different perspectives from which the subjects can be analyzed. Furthermore, the design of the extract-transform-load (ETL) processes responsible for managing the data flow from the sources towards the DW constructs, must also be considered.

Complex business plans and dynamic, evolving enterprise environments often result in a continuous flow of new information requirements that may further require new analytical perspectives or new data to be analyzed. Due to the dynamic nature of the DW ecosystem, building the complete DW design at once is not practical. Also, assuming that all information and business requirements are available from the beginning and remain intact is not realistic either. At the same time, for constructing a DW design –i.e., its MD schema– the heterogeneity and relations among the existing data sources need to be considered as well.

The complexity of the monolithic approach for building a DW satisfying all information requirements, has also been largely characterized in the literature as a stumbling stone during the DW projects (e.g., see [9]). As a solution to this problem, the *Data Warehouse Bus Architecture* has been proposed as a step-by-step approach for building a DW [9]. This approach starts from individual data marts (DM) defined over single data sources and continues exploring the common dimensional structures, which these DMs may possibly share. To facilitate this process, a matrix as the one shown in Table 1 is used, which relates information requirements to facts of the selected DMs and dimensions implied by each DM. Such a matrix is used for detecting how the dimensions are shared among different facts and based on that, for combining different DMs into the DW dimensional model. However, such design guidelines assume a tremendous manual effort from the DW architect and hence, they still encounter the problem of burdensome and time-lasting process of translating the end-user's information requirements into the appropriate MD schema design.

In our work, we follow a different approach to DW design with the goal of automating the process of building the MD schema of a DW by incrementally integrating the information requirements into a unified MD schema design.

In practice, information and business requirements may come from different business users and may span various data sources. For each requirement, a data sources' subset may be identified and translated into the valid multidimensional context that corresponds to the analytical need of such a requirement. Various approaches have both manually and automatically tackled the issue of producing an MD design

Table 1: The DW Bus Architecture for IR1-IR5

	Customer	Supplier	Nation	Region	Orders	Part	Partsupp	Lineitem_dim
ship. qty.(IR1)	√	√	√		√		√	
profit(IR2)		√	√					√
revenue(IR3)		√	√	√		√	√	
avil. stock val.(IR4)		√	√			√		
ship. prior.(IR5)	√				√			√

from data sources, taking into account the end-user requirements. Our method is generic and does not depend on the approach used to accomplish such task.

In a previous effort, we have worked on automating the translation of a single information requirement into the appropriate MD interpretation of a subset of the sources [16] (i.e., interpreting sources' concepts as either factual or dimensional). In addition, such MD interpretations undergo the process of MD validation to ensure the satisfaction of the MD integrity constraints, like those discussed in [10]. However, such MD interpretations as is cannot serve as a single, unified MD schema.

In this paper, we start from such MD interpretations of individual requirements and present an iterative method that consolidates the various MD interpretations (that resemble the rows of Table 1) into a single, unified MD schema design that satisfies the entire set of information requirements.

Our method, *ORE*, is useful for the early stages of a DW project, where we need to create an MD schema design from scratch, but it can also serve during the entire DW lifecycle to accommodate potential evolution events. (As we discuss later on, in the presence of a new or changed requirement, our method does not require creating an MD design from scratch, rather it can automatically absorb the change and fix an existing MD schema.) In both cases, *ORE* processes the MD interpretations of the sources for a single information requirement and aims at producing the complete MD schema satisfying all the requirements so far, while also enabling the minimal MD design by fusing the adjacent facts and dimensions and hiding irrelevant concepts. To achieve better integration into existing MD structures (facts and dimensional hierarchies), *ORE* benefits from the semantics and relations of data sources as these are captured by an appropriate ontology; e.g., synonyms, functional dependencies, taxonomies, and so on.

Outline. The rest of the paper is structured as follows. Section 2 presents an abstract overview of our approach through an example case based on the TPC-H schema [1]. Section 3 formally presents our method for integrating new or changed information requirements into an MD schema design. Finally, Section 4 and Section 5 discuss related work and conclude the paper, respectively.

2. OVERVIEW OF OUR APPROACH

In this section, we first describe an example case and then, we present an overview of our iterative approach to create an MD design from a set of MD interpretations each one corresponding to a single information or business requirement.

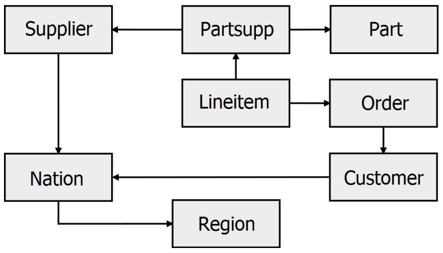

Figure 1: TPC-H Schema

2.1 Example and Background

Our example scenario is based on the TPC-H schema [1], an abstraction of which is illustrated in Figure 1. For the sake of the example, let us assume a set of five information requirements related to the TPC-H schema. We first discuss how we translate these requirements into appropriate MD interpretations (a task which is the focus of a previous work [18]) and then, we discuss how we consolidate these individual MD interpretations into a single, unified MD schema (a task which is the focus of this paper). Our example information requirements are as follows:

- IR1: The total *quantity* of the *parts* shipped from Spanish *suppliers* to French *customers*.
- IR2: For each *nation*, the *profit* for all supplied parts, *shipped* after 01/01/2011.
- IR3: The total *revenue* of the *parts* supplied from East Europe.
- IR4: For German *suppliers*, the total *available stock value* of supplied *parts*.
- IR5: *Shipping priority* and total potential *revenue* of the parts *ordered* before certain *date* and *shipped after certain date* to a *customer* of a given *segment*.

In addition to the information requirements, we use a domain ontology that corresponds to the TPC-H data stores. There are several methods for creating such an ontology (e.g., see [20]) and thus, due to space considerations, we do not elaborate on this topic in this paper. Having such an ontology at hand, we benefit from the ontology features during the integration process.

Starting from the given set of information requirements and data sources, we may map these to a domain ontology for obtaining the corresponding *MD interpretations* of these requirements [16]. Figure 2 illustrates the MD interpretations for each example requirements IR1, ..., IR5.

These interpretations include exactly the subset of source concepts labeled with the appropriate MD roles (i.e., factual, dimensional) to fetch the required data. This means that besides the concepts explicitly related to a requirement, an interpretation may also include concepts relating to the sources (*intermediate concepts*) in order to correctly fetch the data needed. Therefore, for example, even though IR1 does not explicitly require data for customer's *orders*, the corresponding MD interpretation of the sources for IR1 (see Figure 2) does include the `Orders` concept, since this is the only way to relate the `Lineitem` instances with their corresponding `Customers`.

Taking into account the analytical nature of the given information requirements, which typically include one or more concepts to be analyzed –*facts* (e.g., `revenue`, `profit`)– considering different perspectives –*dimensions* (e.g., `Supplier`,

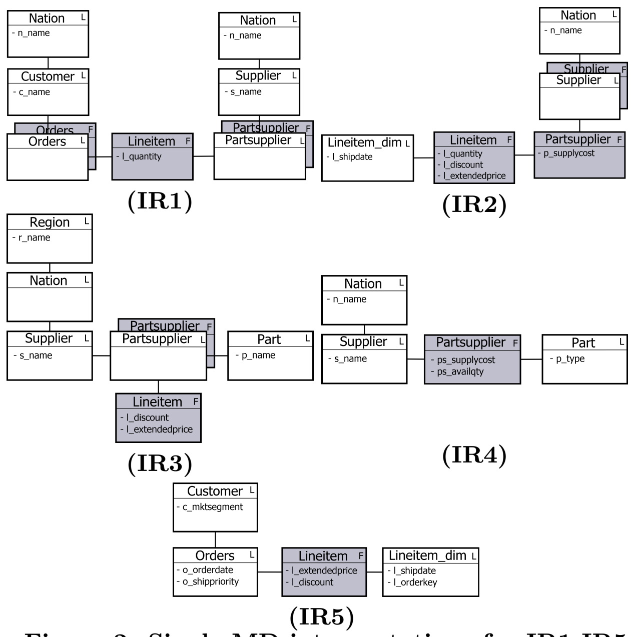
Figure 2: Single MD interpretations for IR1-IR5

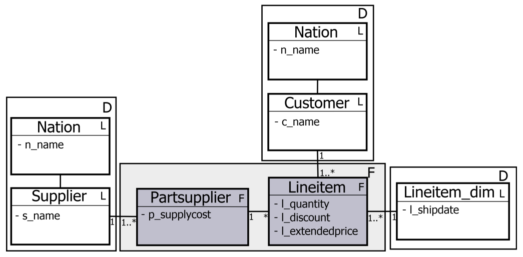
Figure 3: MD schema satisfying IR1&IR2

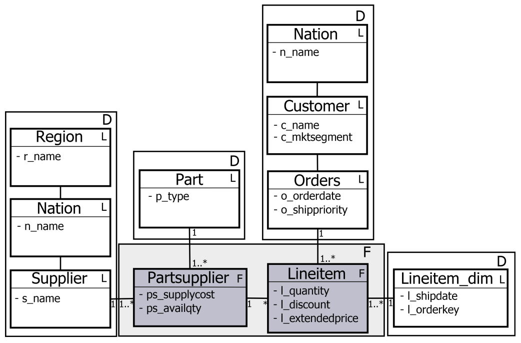
Figure 4: MD schema satisfying IR1-IR5

Customer, Nation)– a given interpretation may be labeled with the corresponding MD information (i.e., source concepts may be either factual or dimensional). For each requirement, we obtain the valid MD-like structure that satisfy the analytical needs of this requirement. Nevertheless, these resulting MD interpretations cannot be considered yet as the final MD schemas, since they may contain unnecessary and/or ambivalent knowledge (e.g., intermediate concepts).

We distinguish two kinds of MD concepts, namely dimensional (L) and factual (F) concepts; graphically these are represented as white and gray rectangles, respectively. A concept may contain MD attributes (measures and descriptors), which are identified in the domain ontology from the input requirements. A concept may have two MD roles (factual and dimensional), which in turn are shown in the MD interpretation of a requirement.

In terms of our example, in the IR2 requirement (see Figure 2), the attributes `l_quantity`, `l_discount`, and `l_extendedprice` are identified as measures, while the attribute `l_shipdate` is identified as dimension. Hence, a new dimensional concept `Lineitem_dim` is created and added to the final MD interpretation for IR2. Furthermore, dimension attribute (`l_shipdate`) is relocated from `Lineitem` into `Lineitem_dim` concept and `Lineitem` may then be tagged with a factual MD role.

2.2 Problem

Starting from a set of input requirements and data sources, we create a set of MD interpretations, one for each requirement. The problem at hand is to identify the common MD knowledge among these interpretations and incrementally integrate them, in order to obtain a single MD schema satisfying all information requirements.

Example. Figure 2 shows the MD interpretations corresponding to the example requirements IR1, ..., IR5. Having as inputs these MD interpretations, we first integrate those obtained for IR1 and IR2 and generate an MD schema satisfying both IR1 and IR2 (see Figure 3). Then, iteratively we integrate the remaining MD interpretations. Finally, we produce an MD schema satisfying all input requirements, as shown in Figure 4. □

2.3 Our Solution, ORE

This section presents the core components of our method for Ontology-based data warehouse REquirement evolution and integration (*ORE*), which are the inputs to our system and the processing stages of *ORE*.

2.3.1 Inputs

Data sources. To boost the integration of the new information requirements into the final MD schema design, we capture the semantics (e.g., concepts, properties) of the available data sources in terms of an OWL ontology. The use of an ontology allows us to automatically infer relations such as synonyms, functional dependencies, taxonomies, etc. among the concepts. Such ontology features provide more semantically meaningful integration of new information requirements. If an ontology capturing the concepts and properties of the data sources is not available, we create it as described in the literature (e.g., [20]).

Multidimensional interpretations (*MDI*). The information requirements posed throughout the organization (similar to the example ones presented in Section 2.1) are validated against the available internal or external sources. The source subset satisfying the given requirements is interpreted with the identified MD knowledge (e.g., as in [16]). Hence, we consider as inputs the MD interpretations satisfying the given set of information requirements (see Figure 2). A single information requirement may be mapped to different MD interpretations due to possible ambivalence of the MD knowledge of the source concepts obtained from this requirement (e.g., intermediate concepts).

For example, for the IR1 requirement (see Figure 2) the intermediate concepts `Orders` and `Partsupp` may have two different MD roles (factual and dimensional) and thus, for IR1 there are four different MD interpretations, since all combinations of the MD roles for `Orders` and `Partsupp` are valid.

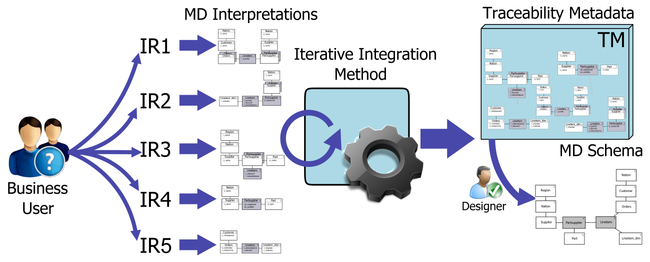

Figure 5: General overview of approach

2.3.2 *Stages*

An abstract schematic flow of our approach is depicted in Figure 5. Our iterative integration method semi-automatically integrates new information requirements and produces the MD schema satisfying the requirements so far. At the same time, the set of operations that illustrates such integration step is identified (i.e., *integration operations*) and further weighted to assist designer's choice. The needed integration operations are shown in Table 2. (Table 2 also contains weights accompanying the integration operations; we discuss this further in Section 3.1.)

Our method, *ORE*, comprises four stages, namely *matching facts*, *matching dimensions*, *complementing the MD design*, and *integration* (see Figure 6). The first three stages gradually match different MD concepts and explore new design alternatives. The last stage considers these matchings and designer's feedback to generate the final MD schema that accommodates a new or changed information requirement. Here, we give a high-level description of these stages and in the next section, we formally present our method.

In all stages, we keep and maintain a structure, namely *traceability metadata* (TM), for systematically tracing everything we know about the MD design integrated so far, like candidate improvements and alternatives. With TM, we avoid overloading the produced MD schema itself. For example, this structure keeps the information about *all* alternative MD interpretations for a single information requirement, while *only one* is chosen to be included in the final MD schema. Iteratively, the traceability metadata grows with the integration of each requirement and, along with the user feedback, it provides the basis for obtaining the final *MD schema*.

Stage 1: Matching facts. We first search for different possibilities of how to incorporate an information requirement (i.e., its MD interpretation) to TM. The matching between factual concepts is considered –i.e., the system searches the fact(s) of TM producing an equivalent set of points in the MD space– as the one in the given MD interpretation. Different possibilities to match the factual concepts results with the appropriate sets of integration operations. The costs of these integration possibilities are further weighted and the prioritized list is proposed to the user, who is expected to choose the one most suitable to her needs.

Stage 2: Matching dimensions. After matching the factual concepts of the new MD interpretation and TM, we then conform the dimensions implied by the matched concepts. Different matchings between levels are considered (i.e., "=", "1 - 1", "1 - *" and "* - 1") and thus, the different valid conformation possibilities are obtained. With each possibility, a different set of integration operations for conforming these dimensions is considered and weighted. The designer decides on what the next integration action(s) should be.

Stage 3: Complementing the MD Design. We further explore the domain ontology and search for new analytical perspectives. Different options may be identified to extend the current schema with new MD concepts (i.e., *levels*, *descriptors*, and *measures*). The designer is then asked to (dis)approve the integration of the discovered concepts into the final MD schema.

Stage 4: Integration. The MD schema is finally obtained in two phases of this stage. First, possible groupings of the adjacent concepts containing the equivalent MD knowledge is identified to minimize the MD design. Both the matchings identified through the previous stages and user feedback are considered, in order to incorporate new requirements into the final MD schema and possibly extend it with the MD knowledge discovered in the ontology and approved by the user. Furthermore, the final MD schema is produced by folding the knowledge and collapsing grouped concepts from previously upgraded TM, to capture only the minimal information relevant to the user. Nevertheless, the complete TM is still preserved in the background to assists further integration steps (e.g., to handle evolution of requirements).

3. THE ORE APPROACH

In this section, we formally present *ORE*, our approach to integration of MD interpretations for individual information requirements. After we formalize the problem, we formally describe each stage of *ORE*.

3.1 Notation and Formalization

Each input information requirement (IR) results into one or more MD interpretations (MDI) of the sources. Along with that, for each requirement the choices made by the user during the integration of that requirement are stored in an ordered list γ, containing the selected ontological relationships. Formally:

$$IR = \{MDI_i | i = 1, ..., n_1\} \cup \gamma$$

The iterative integration method starts from a current MD schema (it can be null at the beginning), whose details are captured in TM. At any given moment, TM contains the set of schemas satisfying IRs so far. Without loss of generality, for the sake of presentation, let us assume that these are star schemas (e.g., snowflakes could also be generated). A single star schema (S) is obtained by integrating one or more information requirements. Formally:

$$S = \{IR_i | i = 1, ..., n_2\}$$

The traceability metadata are formally defined as:

$$TM = \{S_i | i = 1, ..., n_3\}$$

TM is also used for handling evolving requirements. When a requirement changes, we update TM (i.e., TM_{new}=TM_{old}-IR_{old} + IR_{new}) and generate a new MD schema, taking into account previously registered user feedback as stored into γ.

In each iteration, our method semi-automatically identifies a set of *integration operations* necessary to incorporate the new IR into the current MD schema obtained from TM. The complete set of integration operations is given in Table 2. Each operation comes with a weight representing the significance of the operation. These weights are used for prioritizing the alternative matching options before presenting

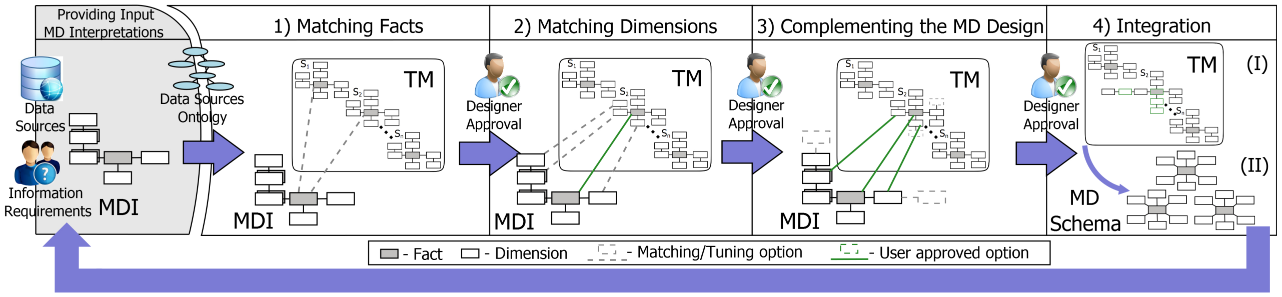

Figure 6: *ORE*: iterative integration method

Table 2: Integration operations

Operation name	Weight
insertLevel	0,21
insertFact	0,31
insertRollUpRelation	0,27
renameConcept	0
insertDimDescriptor	0,04
insertFactMeasure	0,36
MergeLevels	0,04×(♯insertDimDescriptor)
MergeFacts	0,36×(♯insertFactMeasure)

them to the designer (see also Section 3.3). The weights refer to the correlation between DW quality factors (e.g., the structural complexity, understandability, analizability, maintainability, etc.) and different structural characteristics (e.g., *number of dimensions*, *number of facts*, *number of dimensional/factual attributes*, *number of functional dependencies*, etc.), according to [19]. However, note that the choice of their values is orthogonal to our approach.

3.2 Pre-processing

While matching the different concepts or exploring new analytical perspectives, we take into account the characteristics and relationships among the concepts captured by the domain ontology. This allows a more effective integration of the heterogeneous hierarchies and factual concepts by exploiting different ontology features, e.g., taxonomies, many-to-one (i.e., "* - 1"), one-to-many (i.e., "1 - *"), and one-to-one or synonym (i.e., "1 - 1") relationships. Note, that we do not explore many-to-many ("* - *") relationships because such associations violate MD integrity constraints and thus, they cannot be considered for MD design [10]. To automate the search for different relationships among the concepts we take advantage of the ontology inference engine. At the same time, the transitive closure of the different relationships ("1 - 1", "1 - *" or "* - 1") in the ontology is considered. As it has been shown in the past, it is feasible to automate the search of such closures for discovering functional dependencies in the ontology [17]. To enhance the performance of our iterative method, we consider pre-computing the transitive closure of functional dependencies for the concepts currently present in the TM and the ones coming with new MDI.

3.3 Matching facts

ORE starts from the MDIs obtained for a given information requirement and it explores TM to find the existing star schema(s) $S_i \in TM$ with the *matching facts*. To match two facts, the condition that these facts should produce an equivalent set of points in the MD space must be satisfied. This condition is formally defined as follows:

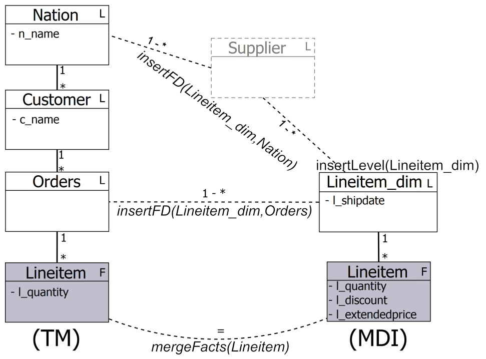

Figure 7: Matching the MD Interpretation for IR2

(C_1): The fact $F_{mdi} \in MDI$ *matches* the fact $F_{si} \in S_i$ iff there is a *bijective function* f such that for each point x_{mdi} in the MD space arranged by the dimensions $\{d_1, d_2, .., d_n\}$ implied by F_{mdi}, there is one and only one point y_{si} in the MD space arranged by the dimensions $\{d'_1, d'_2, .., d'_m\}$ implied by F_{si}, such that $f(x_{mdi}) = y_{si}$.

Our method tests whether C_1 is satisfied (i.e., that F_{mdi} matches F_{si}) by means of identifying that at least one of the two following properties between F_{mdi} and F_{si} holds:

(P_1) F_{mdi} is related to F_{si} by means of "1 - 1" correspondence (i.e., synonyms or equal factual concepts).

(P_2) For each dimension d_i (d'_i) of F_{mdi} (F_{si}) there is the dimension d'_j (d_j) of F_{si} (F_{mdi}), such that d_i and d'_j (d'_i and d_j) are related by means of "1 - 1" relationship or d_i (d'_i) is related to F_{si} (F_{mdi}) by means of "1 - *" relationship.

Example (P_1). Figures 7 and 8 show how the matching schema in TM is identified for MDIs produced for information requirements $IR2$ and $IR5$, respectively, by means of identified "1 - 1" correspondence of their facts `Lineitem`. □

Example (P_2). Figure 9 shows an example inspired by TPC-H. A new `PartMarket` fact is introduced, which assesses the convenience of releasing a specific `Part` in a specific market (i.e., `Nation`). Even though we may find no "1 - 1" correspondence between `PartMarket` and the `Supplier-PartSupplier` fact, their MD spaces do coincide as they are arranged by the same dimensions: `Nation` and `Part`. □

For each $F_{si} \in S_i$ matched with $F_{mdi} \in MDI$, the *mergeFacts(F_{si}, F_{mdi})* operation is identified with its corresponding weight (see Table 2). If an equality between the given facts is not identified, the *renameConcept(F_{si}, F_{mdi})* is also added. If for F_{mdi} our method does not identify a valid matching in TM, the *insertFact*(F_{mdi}) operation is considered. All integration possibilities are listed with their corresponding weights and user feedback is required to continue.

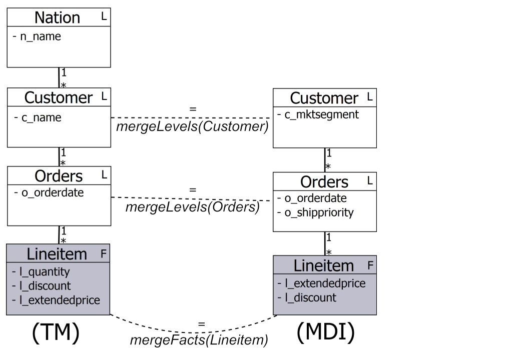

Figure 8: Matching the MD Interpretation for IR5

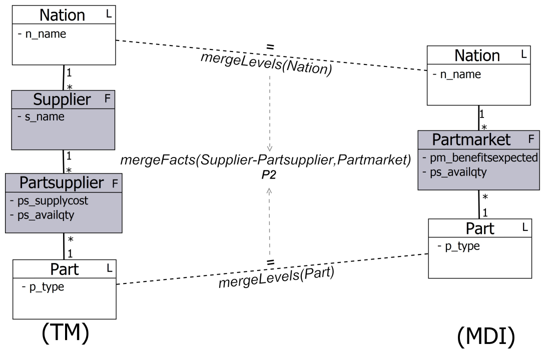

Figure 9: Matching facts *(P2)*

3.4 Matching dimensions

For each pair of matching facts (i.e., F_{mdi}, F_{si}) identified in the previous stage and chosen by the user, *ORE* continues by conforming their dimensions. To this end, *ORE* in this stage identifies possible matchings of the dimensional concepts of MDI inside the corresponding star schema S_i of TM (i.e., $F_{si} \in S_i$).

Since a single dimension d_i consists of a partially ordered set of individual levels, with single bottom (i.e., atomic) level and with "to-one" relationships among the levels, each dimension may be seen as a directed acyclic graph (DAG), where the vertices of the graph are individual levels and the directed edges between them "$\rightarrow$" are "to-one" relationships. Note that we assume the graph is guaranteed to be acyclic in a preliminar validation step of the requirement, since the loops would violate the MD integrity constraints [10].

To match each pair of levels, we consider the *minimum path* of a valid MD relation (=, "1-1", "1-*" or "*-1") between them, i.e., the path relating two level concepts of different dimensions in the ontology and not including any other level of these dimensions.

Example. In Figure 7, while we identify "* - 1" matchings from `Lineitem_dim` to `Orders` and `Nation`, we do not consider the one from `Lineitem_dim` to `Customer` since it goes over `Orders`. □

For each dimension of MDI we search for a compatible dimension in S_i; i.e., the dimension with atomic levels related with some kind of valid MD relation (=, "1-1", "1-*" or "*-1"). For these two dimensions we search for possible matchings among their individual levels. The problem may be seen as a graph matching problem. Next, we present the dimension-matching, DM algorithm, which solves the problem in our context. The algorithm, for two dimensions coming from MDI and S_i, represented with their corresponding DAGs (i.e., D_{mdi} and D_{si}), searches for the set of operations ($intOps$) that are necessary to be applied over S_i inside TM to appropriately integrate the new information requirement (i.e., its corresponding MDI).

Considering the topological order of the nodes in D_{mdi} and D_{si}, the algorithm recursively explores the paths in D_{mdi} and D_{si} starting from their atomic levels. In each recursive call, the matching between two levels and their remaining hierarchies is considered. If there is no relation between two levels, DM stops searching for matchings in their hierarchies. In doing so, DM discards unrelated parts of the dimensions and thus, prunes the search space.

Otherwise, following the topological order (Step 2c), DM keeps searching for different possibilities to relate the levels and collects the corresponding operations along the way. In each recursive call, we consider the next two levels from the topological orders of D_{mdi} and D_{si}, $curLevelD_{mdi}$ and $curLevelD_{si}$, respectively (Step 2). If we can infer the full match between $curLevelD_{mdi}$ and $curLevelD_{si}$ (Step 2a), i.e., equality or synonyms, we consider the *mergeLevels* operation to be applied. Additionally, in the case of synonyms, our method potentially considers the *renameConcept* operation over the $curLevelD_{mdi}$ (Step 2(a)ii).

Example. Figure 8 shows an integration possibility for the `Orders` dimension, in the case of integrating IR5 into TM satisfying IR1-IR4. A full matching (i.e., equality) is then found for both `Orders` and `Customer` levels of the corresponding D_{mdi}. Thus, the *mergeLevels* operations are proposed for both `Orders` and `Customer` levels and they respectively involve *insertDimDescriptor* operations for transfering `o_shippriority` and `c_mktsegment` level attributes. □

For "1 - *" or "* - 1" relationships, we consider inserting a new level into the star S_i (i.e., *insertLevel* operation). Additionally, for "1 - *" or "* - 1" relationships (Steps 2(b)ii and 2(b)iii), we consider inserting a roll-up relation, i.e., functional dependency, (i.e., *insertRollUpRelation* operation).

Example. In Figure 7, for all "* - 1" or "1 - *" matchings found, insertion of a roll-up relation is proposed; e.g., `insertRollUpRelation(Lineitem_dim, Orders)`. □

Furthermore, we continue matching the dimensions by considering different combinations of the remaining hierarchies starting from the current levels (i.e., $curLevelD_{mdi}$ and $curLevelD_{si}$) (Step 2d).

Similarly to the previous stage, we may identify different possibilities to conform the dimensions between MDI and S_i, and therefore, these possibilities are weighted based on the set of integration operations they imply and proposed to the user. User feedback is then expected to determine how to conform the given dimensions.

3.5 Complementing the MD design

This stage considers the possible integration of the new MDI in TM, identified in the previous stages, and further explores the ontology to complement the future MD design with new analytically interesting concepts. By exploring the functional dependencies ("to-one" relationships) in the ontology, new *levels* for the previously conformed dimensions may be identified. Furthermore, different *datatype properties* in the ontology may also be identified either as *measures* of the existing facts or *descriptive attributes* of the levels. We distinguish two cases:

- If the property has numerical datatype (e.g., *integer*, *double*), we consider using it as a *measure* iff the domain concept of the property is identified as a fact.

Algorithm: DM

```
inputs: $D_{mdi}$, $curLevelD_{mdi}$, $D_{si}$, $curLevelD_{si}$, output: $intOps$

1. $intOps := \emptyset$;
2. If related($curLevelD_{mdi}$, $curLevelD_{si}$) then
   (a) If $curLevelD_{mdi}$ fully matches $curLevelD_{si}$ then
       i. $intOps$ := add($intOps$, mergeLevels($curLevelD_{mdi}$,$curLevelD_{si}$) );
       ii. If multiplicity($curLevelD_{mdi}$,$curLevelD_{si}$) = "1 - 1" then
           $intOps$ := add($intOps$,renameConcept($curLevelD_{si}$,$curLevelD_{mdi}$));
   (b) else
       i. $intOps$ := add($intOps$, insertLevel($curLevelD_{mdi}$));
       ii. If multiplicity($curLevelD_{mdi}$,$curLevelD_{si}$) = "1 - *" then
           $intOps$ := add($intOps$, insertRollUpRelation($curLevelD_{si}$,$curLevelD_{mdi}$));
       iii. If multiplicity($curLevelD_{mdi}$,$curLevelD_{si}$) = "* - 1" then
           $intOps$ := add($intOps$, insertRollUpRelation($curLevelD_{mdi}$,$curLevelD_{si}$));
   (c) $levelsD_{mdi}$ := getTopOrderLevels($D_{mdi}$, $curLevelD_{mdi}$); $levelsD_{si}$ := getTopOrderLevels($D_{si}$, $curLevelD_{si}$);
   (d) Foreach pair($nextLevelD_{mdi}$ from getNext($levelsD_{mdi}$), $nextLevelD_{si}$ from getNext($levelsD_{si}$)) do
       i. $intOps$ := add($intOps$, $DM$($D_{mdi}$, $nextLevelD_{mdi}$, $D_{si}$, $nextLevelD_{si}$));
3. return $intOps$;
```

- Otherwise, in the case that the domain concept of a property is identified as a level, our method suggests using the property as a new *descriptor*.

Different possibilities for enriching the current design are presented to the designer as different integration operations (i.e., *insertLevel*, *insertFactMeasure*, *insertDimDescriptor*). The designer may decide how to complement the MD design.

Example. For the `Orders` dimension in Figure 8, we explore the ontology and propose concept `Region` as a new top level of the given dimension. Also, we propose different descriptors for the levels `Ordes`, `Customer`, `Nation`, and `Region`; e.g., `o_orderstatus`, `c_phone`, and so on. □

3.6 Integration

Having identified the matching of concepts and the options for complementing the MD design in the previous stages, in this stage, *ORE* produces the final MD schema in a two-phase process.

(I) Partitioning. The first phase starts from the MD schemas obtained in TM, so far. Each MD schema as it can be represented with a directed acyclic graph (DAG) is further partitioned into different groups of concepts (subgraphs), so that all concepts of one group:

1. Produce a connected subgraph and
2. Have the same MD interpretation (i.e., all concepts are either factual of dimensional).

Each of these subgraphs of concepts further gives rise to a fact or a dimension of the final star schema.

(II) Folding. Starting from these partitions we obtain the final star schema (or any other schema like a snowflake schema). Inside each subgraph captured by a single partition, we consider only the concepts currently required by the user, either provided with the requirement at hand or discovered when complementing the MD design in the ontology (i.e., Stage 3). The concepts considered inside each partition are then collapsed to produce an element (i.e., fact or dimension) of the final MD schema.

In these two phases, first, we relax the final schema from knowledge currently irrelevant to the designer's choices and then, we conform the schema to well-known DW quality factors (e.g., the structural complexity, understandability, analizability, maintainability, and others as discussed in [19]). For example, collapsing the adjacent levels simplifies the corresponding dimensions and lowers the number of roll-up relationships, which has a significant influence in the overall structural complexity of the schema.

While concepts with currently no interest to the designer may be hidden from the final MD schema design, TM still preserves all this knowledge for using it in future integration steps. Furthermore, as TM contains the knowledge from data sources that answers the given set of information requirements, we can benefit from it for producing the appropriate data flow design (e.g., ETL flow) to manage the transfer of the data from the sources to the produced target MD schema. We consider this as a possible future work.

4. RELATED WORK

As we discussed in Section 1, following a monolithic approach for building a DW is considered problematic and thus, manual guidelines were given to overcome this issue (e.g., DW Bus Architecture [9]). Apart from traditional DW designing approaches, e.g., [6, 7, 11], various works have studied the problem of adjusting the DW systems to changes of the business environments. For dealing with such an issue, different directions have been followed.

Schema Evolution. The approaches that fall into this category, e.g., [3, 15, 23], have as main goal to maintain the up-to-dateness of the existing DW models when a change occurs. While almost all of them propose a set of evolution operators (e.g., for adding/deletion of dimensions/levels/facts or their instances), some (e.g., [15]) also study the influence of different evolution changes on the quality metrics of the DW (e.g., consistency and completeness). Similar problems have been studied for ETL flows too (e.g., [14]).

Schema Versioning. Beside dealing with the changes by upgrading the existing schema, schema versioning approaches have also focused on keeping the trace of these changes by separately maintaining different versions of the schema (e.g., [2, 4, 5]). Some of them (e.g., [2]) in addition to the versions resulting from real world changes, they also store and maintain the alternative versions which can simulate various operational/business scenarios and propose new analytical perspectives. Another work deals with the problem of cross-version querying and introduces the concept of augmented schema, which keep track of change actions to enable answering the queries spanning the validity of different versions [5].

One contribution of these works is the formal definition of specific alternation operators, which can be applied when an evolution change occurs. However, these works do not study how the information requirements actually affect such evolution changes. In this sense, our work complements them and starting from a given set of information requirements, it aims to automatically derive the changes of the current schema necessary to be applied for satisfying these new requirements.

Incremental DW Design and DW Schema Integration. There are works that have studied the problem of incremental DW design; e.g., [13, 21]. For example, a previous work considers the DW as a set of materialized views over the source relations and introduces a state space search algorithm for maintaining this set (i.e., *view maintenance*) to answer new queries [21]. Since a pure relational approach has been followed, it uses equivalence transformation rules to enhance the integration of new queries into the existing view set. That makes this approach not easily applicable to the current heterogeneous environments. In fact, as it has been become clear in the last decade, DW are more complex than just a set of materialized views, especially when the ETL flows come into play. On the schema integration side, there are works that use ontologies, to bridge the semantic gap among heterogeneous data (e.g., [20]). To deal with the integration of heterogeneous DW schemas, another work proposes two approaches: loosely and tightly coupled [22]. But, this work assumes that a structural matching among schemas exists and proposes the d-chase procedure (inspired by the chase procedure) for the chase of dimensional instances to ensure the consistency of the given matching.

Overall, the importance of information requirements into the DW design process has been generally overlooked. An exception is the work in [12] that starts from OLAP requirements expressed in sheet-style tables and later translates them to single star schemas. However, the integration proposed is based solely on union operations applied to the facts and dimension hierarchies. Moreover, this work does not consider the heterogeneity and complex relations that may exist in such structures.

5. CONCLUSIONS

Our research aims at providing an end-to-end, requirement-driven solution for designing MD schemata and ETL flows for the DW ecosystem. In the past, we presented our system called GEM, which translates information and business requirements into MD interpretations of these requirements and also, produces a set of necessary ETL operations for managing the data flow from the sources and answering the given requirements [18]. GEM produces separate ETL and MD designs per requirement. In a recent work, we have shown how to integrate those individual ETL designs to a single ETL model [8]. In this paper, we have extended GEM to support the production of a single, unified MD schema that fulfills all input requirements. We have discussed the challenges and have presented an iterative method for consolidating MD interpretations, through an example case.

There are many interesting future directions. A prominent one is to explore the functionality achieved and technology produced in this work, for fine-tune the data flow design (e.g., ETL).

6. ACKNOWLEDGEMENTS

This work has been partly supported by the Spanish Ministerio de Ciencia e Innovación under project TIN2011-24747.

7. REFERENCES

[1] TPC-H, http://www.tpc.org/tpch/
[2] Bebel, B., Eder, J., Koncilia, C., Morzy, T., Wrembel, R.: Creation and management of versions in multiversion data warehouse. In: SAC. pp. 717–723 (2004)
[3] Blaschka, M., Sapia, C., Höfling, G.: On schema evolution in multidimensional databases. In: DaWaK. pp. 153–164 (1999)
[4] Body, M., Miquel, M., Bédard, Y., Tchounikine, A.: A multidimensional and multiversion structure for OLAP applications. In: DOLAP. pp. 1–6 (2002)
[5] Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. Data Knowl. Eng. 59(2), 435–459 (2006)
[6] Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst. 7(2-3), 215–247 (1998)
[7] Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual data warehouse modeling. In: DMDW. p. 6 (2000)
[8] Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Integrating ETL processes from information requirements. In: DaWaK. pp. 65–80 (2012)
[9] Kimball, R., Reeves, L., Thornthwaite, W., Ross, M.: The Data Warehouse Lifecycle Toolkit. J. Wiley & Sons (1998)
[10] Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modeling. Data Knowl. Eng. 68(12), 1452–1469 (2009)
[11] Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems 45(1), 41–58 (2008)
[12] Nabli, A., Feki, J., Gargouri, F.: Automatic construction of multidimensional schema from OLAP requirements. In: AICCSA. p. 28 (2005)
[13] Naggar, P., Pontieri, L., Pupo, M., Terracina, G., Virardi, E.: A model and a toolkit for supporting incremental data warehouse construction. In: DEXA. pp. 123–132 (2002)
[14] Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: Policy-regulated management of ETL evolution. J. Data Semantics 13, 147–177 (2009)
[15] Quix, C.: Repository support for data warehouse evolution. In: DMDW. p. 4 (1999)
[16] Romero, O., Abelló, A.: Automatic Validation of Requirements to Support Multidimensional Design. Data Knowl. Eng. 69(9), 917–942 (2010)
[17] Romero, O., Calvanese, D., Abelló, A., Rodriguez-Muro, M.: Discovering functional dependencies for multidimensional design. In: DOLAP. pp. 1–8 (2009)
[18] Romero, O., Simitsis, A., Abelló, A.: GEM: Requirement-driven generation of ETL and multidimensional conceptual designs. In: DaWaK. pp. 80–95 (2011)
[19] Serrano, M.A., Calero, C., Piattini, M.: An experimental replication with data warehouse metrics. IJDWM 1(4), 1–21 (2005)
[20] Skoutas, D., Simitsis, A.: Ontology-based conceptual design of ETL processes for both structured and semi-structured data. Int. J. Semantic Web Inf. Syst. 3(4), 1–24 (2007)
[21] Theodoratos, D., Sellis, T.K.: Incremental design of a data warehouse. J. Intell. Inf. Syst. 15(1), 7–27 (2000)
[22] Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distributed and Parallel Databases 23(1), 69–97 (2008)
[23] Vaisman, A.A., Mendelzon, A.O., Ruaro, W., Cymerman, S.G.: Supporting dimension updates in an OLAP server. Inf. Syst. 29(2), 165–185 (2004)