An Automatic Schema-Instance Approach for Merging Multidimensional Data Warehouses

Yuzhao Yang
IRIT-CNRS (UMR 5505), Université de Toulouse
Toulouse, France
yuzhao.yang@irit.fr

Jérôme Darmont
Université de Lyon, Lyon 2, UR ERIC
Lyon, France
jerome.darmont@univ-lyon2.fr

Franck Ravat
IRIT-CNRS (UMR 5505), Université de Toulouse
Toulouse, France
franck.ravat@irit.fr

Olivier Teste
IRIT-CNRS (UMR 5505), Université de Toulouse
Toulouse, France
olivier.teste@irit.fr

ABSTRACT

Using data warehouses to analyse multidimensional data is a significant task in company decision-making. The need for analyzing data stored in different data warehouses generates the requirement of merging them into one integrated data warehouse. The data warehouse merging process is composed of two steps: matching multidimensional components and then merging them. Current approaches do not take all the particularities of multidimensional data warehouses into account, e.g., only merging schemata, but not instances; or not exploiting hierarchies nor fact tables. Thus, in this paper, we propose an automatic merging approach for star schema-modeled data warehouses that works at both the schema and instance levels. We also provide algorithms for merging hierarchies, dimensions and facts. Eventually, we implement our merging algorithms and validate them with the use of both synthetic and benchmark datasets.

CCS CONCEPTS

• **Information systems → Data warehouses**.

KEYWORDS

Multidimensional data warehouse, schema-instance merging, automatic integration

ACM Reference Format:
Yuzhao Yang, Jérôme Darmont, Franck Ravat, and Olivier Teste. 2021. An Automatic Schema-Instance Approach for Merging Multidimensional Data Warehouses. In *25th International Database Engineering & Applications Symposium (IDEAS 2021), July 14–16, 2021, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3472163.3472268

1 INTRODUCTION

Data warehouses (DWs) are widely used in companies and organizations as an important Business Intelligence (BI) tool to help build decision support systems [9]. Data in DWs are usually modeled in a multidimensional way, which allows users to consult and analyze the aggregated data through multiple analysis axes with On-Line Analysis Processing (OLAP) [14]. In a company, various independent DWs containing some common elements and data may be built for different geographical regions or functional departments. There may also exist common elements and data between the DWs of different companies. The ability to accurately merge diverse DWs into one integrated DW is therefore considered as a major issue [8]. DW merging constitutes a promising solution to provide more opportunities of analysing the consistent data coming from different sources.

A DW organizes data according to analysis subjects (facts) associated with analysis axes (dimensions). Each fact is composed of indicators (measures). Finally, each dimension may contain one or several analysis viewpoints (hierarchies). Hierarchies allow users to aggregate the attributes of a dimension at different levels to facilitate analysis. Hierarchies are identified by attributes called parameters.

Merging two DWs is a complex task that implies solving several problems. The first issue is identifying the common basic components (attributes, measures) and defining semantic relationships between these components. The second issue is merging schemata that bear common components. Merging two multidimensional DWs is difficult because two dimensions can (1) be completely identical in terms of schema, but not necessarily in terms of instances; (2) have common hierarchies or have sub-parts of hierarchies in common without necessarily sharing common instances. Likewise, two schemata can deal with the same fact or different facts, and even if they deal with the same facts, they may or may not have measures in common, without necessarily sharing common data.

Moreover, a merged DW should respect the constraints of the input multidimensional elements, especially the hierarchical relationships between attributes. When we merge two dimensions having matched attributes of two DWs, the final DW should preserve all the partial orders of the input hierarchies (i.e., the binary aggregation relationships between parameters) of the two dimensions. It is also necessary to integrate all the instances of the input DWs, which may cause the generation of empty values in the merged DW. Thus, the merging process should also include a proper analysis of empty values.

In sum, the DW merging process concerns matching and merging tasks. The matching task consists in generating correspondences between similar schema elements (dimension attributes and fact measures) [4] to link two DWs. The merging task is more complex and must be carried out at two levels: the schema level and the instance level. Schema merging is the process of integrating several schemata into a common, unified schema [12]. Thus, DW schema merging aims at generating a merged unified multidimensional schema. The instance level merging deals with the integration and management of the instances. In the remainder of this paper, the term "matching" designates schema matching without considering instances, while the term "merging" refers to the complete merging of schemata and corresponding instances.

To address these issues, we define an automatic approach to merge two DWs modeled as star schemata (i.e., schemata containing only one fact table), which (1) generates an integrated DW conforming to the multidimensional structures of the input DWs, (2) integrates the input DW instances into the integrated DW and copes with empty values generated during the merging process.

The remainder of this paper is organized as follows. In Section 2, we review the related work about matching and merging DWs. In Section 3, we specify an automatic approach to merge different DWs and provide DW merging algorithms at the schema and instance levels. In Section 4, we experimentally validate our approach. Finally, in Section 5, we conclude this paper and discuss future research.

2 RELATED WORK

DW merging actually concerns the matching and the merging of multidimensional elements. We classify the existing approaches into four levels: matching multidimensional components, matching multidimensional schemata, merging multidimensional schemata and merging DWs.

A multidimensional component matching approach for matching aggregation levels is based on the fact that the cardinality ratio of two aggregation levels from the same hierarchy is nearly always the same, no matter the dimension they belong to [3]. Thus, by creating and manipulating the cardinality matrix for different dimensions, it is possible to discover the matched attributes.

The matching of multidimensional schemata directs at discovering the matching of every multidimensional components between two multidimensional schemata. A process to automatically match two multidimensional schemata is achieved by evaluating the semantic similarity of multidimensional component names [2]. Attribute and measure data types are also compared in this way. The selection metric of bipartite graph helps determine the mapping choice and define rules aiming at preserving the partial orders of the hierarchies at mapping time. Another approach matches a set of star schemata generated from both business requirements and data sources [5]. Semantic similarity helps find the matched facts and dimension names. Yet, the DW designer must intervene to manually identify some elements.

A two-phase approach for automatic multidimensional schema merging is achieved by transforming the multidimensional schema into a UML class diagram [7]. Then, class names are compared and the number of common attributes relative to the minimal number of attributes of the two classes is computed to decide whether two classes can be merged.

DW merging must operate at both schema and instance levels. Two DW merging approaches are the intersection and union of the matched dimensions. Instance merging is realized by a d-chase procedure [15]. The second merging strategy exploits similar dimensions based on the equivalent levels in schema merging [11]. It also uses the d-chase algorithm for instance merging. However, the two approaches above do not consider the fact table. Another DW merging approach is based on the lexical similarity of schema string names and instances, and by considering schema data types and constraints [8]. Having the mapping correspondences, the merging algorithm takes the preservation requirements of the multidimensional elements into account, and is formulated to build the final consolidated DW. However, merging details are not precise enough and hierarchies are not considered.

To summarize, none of the existing merging methods can satisfy our DW merging requirements. Some multidimensional components are ignored in these approaches, and the merging details of each specific multidimensional components is not explicit enough, which motivates us to propose a complete DW merging approach.

3 PRELIMINARIES

We introduce in this section the basic concepts of multidimensional DW design [13]. The multidimensional DW can be modelled by a star or a constellation schema. In the star schema, there is a single fact connected with different dimensions, while the constellation schema consists of more than one fact which share one or several common dimensions.

Definition 3.1. A constellation denoted C is defined as $(N^C, F^C, D^C, Star^C)$ where N^C is a constellation name, $F^C = \{F_1^C, ..., F_m^C\}$ is a set of facts, $D^C = \{D_1^C, ..., D_n^C\}$ is a set of dimensions, $Star^C : F^C \rightarrow 2^{D^C}$ associates each fact to its linked dimensions. A star is a constellation where F^C contains a single fact; i.e. $m = 1$.

A dimension models an analysis axis and is composed of attributes (dimension properties).

Definition 3.2. A dimension, denoted $D \in D^C$ is defined as (N^D, A^D, H^D, I^D) where N^D is a dimension name, $A^D = \{a_1^D, ..., a_u^D\} \cup \{id^D\}$ is a set of attributes, where id^D represents the dimension identifier, which is also the parameter of the lowest level and called the root parameter. $H^D = \{H_1^D, ..., H_v^D\}$ is a set of hierarchies, $I^D = \{i_1^D, ..., i_p^D\}$ is a set of dimension instances. The value of the instance i_p^D for an attribute a_u^D is annotated as $i_p^D.a_u^D$.

Dimension attributes (also called parameters) are organised according to one or more hierarchies. Hierarchies represent a particular vision (perspective) and each parameter represents one data granularity according to which measures could be analysed.

Definition 3.3. A hierarchy of a dimension D, denoted $H \in H^D$ is defined as $(N^H, Param^H)$ where N^H is a hierarchy name, $Param^H =< id^D, p_2^H, ..., p_v^H >$ is an ordered set of dimension attributes, called parameters, which represent useful graduations along the dimensions, $\forall k \in [1...v], p_k^H \in A^D$. The roll up relationship between two parameters can be denoted by $p_1^H \preceq_H p_2^H$

for the case where p_1^H roll up to p_2^H in H. For $Param^H$, we have $id^D \preceq_H p_1^H, p_1^H \preceq_H p_2^H, ..., p_{v-1}^H \preceq_H p_v^H$. The matching of multidimensional schemata is based on the matching of parameters, the matching relationship between two parameters of two hierarchies $p_i^{H_1}$ and $p_j^{H_2}$ is denoted as $p_i^{H_1} \simeq p_j^{H_2}$.

A sub-hierarchy is a continuous sub-part of a hierarchy which we call the parent hierarchy of the sub-hierarchy. This concept will be used in our algorithms, but it is not really meaningful. So a sub-hierarchy has the same elements than a hierarchy, but its lowest level is not considered as "id". All parameters of a sub-hierarchy are contained in its parent hierarchy and have the same partial orders than those in the parent hierarchy. "Continuous" means that in the parameter set of the parent hierarchy of a sub-hierarchy, between the lowest and highest level parameters of the sub-hierarchy, there is no parameter which is in the parent hierarchy but not in the sub-hierarchy.

Definition 3.4. A sub-hierarchy SH of $H \in H^D$ is defined as $(N^{SH}, Param^{SH})$ where N^{SH} is a sub-hierarchy name, $Param^{SH} = < p_1^{SH}, ..., p_v^{SH} >$ is an ordered set of parameters, called parameters, $\forall k \in [1...v], p_k^H \in Param^H$. According to the relationship between a sub-hierarchy and its parent hierarchy, we have: (1) $\forall p_1^{SH}, p_2^{SH} \in Param^{SH}, p_1^{SH} \preceq_{SH} p_2^{SH} \Rightarrow p_1^{SH}, p_2^{SH} \in Param^H \wedge p_1^{SH} \preceq_H p_2^{SH}$, (2) $\forall p_1^H, p_2^H, p_3^H \in Param^H, p_1^H \preceq_H p_2^H \wedge p_2^H \preceq_H p_3^H \wedge p_1^H, p_3^H \in Param^{SH} \Rightarrow p_2^H \in Param^{SH}$.

A fact reflects information that has to be analysed according to dimensions and is modelled through one or several indicators called measures.

Definition 3.5. A fact, noted $F \in F^C$ is defined as $(N^F, M^F, I^F, IStar^F)$ where N^F is a fact name, $M^F = \{m_1^F, ..., m_w^F\}$ is a set of measures. $I^F = \{i_1^F, ..., i_q^F\}$ is a set of fact instances. The value of a measure m_w^F of the instance i_q^F is denoted as $i_q^F.m_w^F$. $IStar^F : I^F \rightarrow \mathcal{D}^F$ is a function where $\mathcal{D}^F$ is the cartesian product over sets of dimension instances, which is defined as $\mathcal{D}^F = \prod_{D_k \in Star^C(F)} I^{D_k}$. $IStar^F$ associates fact instances to their linked dimension instances.

We complete these definitions by a function $extend(H_1, H_2)$ allowing to extend the parameters of the first (sub)hierarchy H_1 by the other one (H_2).

4 AN AUTOMATIC APPROACH FOR DW MERGING

Like illustrated in Figure 1, merging two DWs implies matching steps and steps dedicated to the merging of dimensions and facts. The matching of parameters and measures are based on syntactic and semantic similarities [10][6] for the attribute or measure names. Since the matching is intensively studied in the literature, we focus in this paper only on the merging steps of our process (green rectangle in Figure 1). In regard to the merging, we firstly define an algorithm for the merging of hierarchies by decomposing two hierarchies into sub-hierarchy pairs and merging them to get the final hierarchy set. Then, we define an algorithm of dimension merging concerning both instance and schema levels and which completes some empty values. Finally, we define an algorithm of the star merging based on the dimension merging algorithm which merges the dimensions and the facts at the schema and instance levels and corrects the hierarchies after the merging.

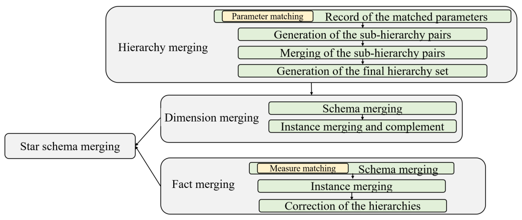

Hierarchy merging
Parameter matching
Record of the matched parameters
Generation of the sub-hierarchy pairs
Merging of the sub-hierarchy pairs
Generation of the final hierarchy set
Star schema merging
Dimension merging
Schema merging
Instance merging and complement
Fact merging
Measure matching
Schema merging
Instance merging
Correction of the hierarchies

Figure 1: Overview of the merging process

4.1 Hierarchy merging

In this section, we define the schema merging process of two hierarchies coming from two different dimensions. The first challenge is that we should preserve the partial orders of the parameters. The second one is how to decide the partial orders of the parameters coming from different original hierarchies. These challenges are solved in the algorithm proposed below which is achieved by 4 steps: record of the matched parameters, generation of the sub-hierarchy pairs, merging of the sub-hierarchy pairs and generation of the final hierarchy set.

Algorithm 1 $MergeHierarchies(H_1, H_2)$

```
Output: A set of merged hierarchies $H'$ or two sets of merged hierarchies $H^{1'}$ and $H^{2'}$
1:  $M, SH', H' \leftarrow \emptyset$; //$M$ is an ordered set of the couples of matched
    parameters with possibly the couple of the last parameters, for the $n$th
    parameter couple $M[n-1]$, $M[n-1][0]$ represents the parameter of
    $H_1$ in $M[n-1]$, while $M[n-1][1]$ represents the one of $H_2$.
2:  $Param^{SH_1}, Param^{SH_{1'}}, Param^{SH_2}, Param^{SH_{2'}} \leftarrow \emptyset$;
3:  for each $p_i^{H_1} \in Param^{H_1}$ do
4:      for each $p_j^{H_2} \in Param^{H_1}$ do
5:          if $p_i^{H_1} \simeq p_j^{H_2}$ then
6:              $M \leftarrow M + < p_i^{H_1}, p_j^{H_2} >$;
7:          end if
8:      end for
9:  end for
10: if $M = \emptyset$ then
11:     $H^{1'} \leftarrow \{H_1\}; H^{2'} \leftarrow \{H_2\}$;
12:     return $H^{1'}, H^{2'}$
13: else
14:     $m_l \leftarrow < Param^{H_1}[|Param^{H_1}|-1], Param^{H_2}[|Param^{H_2}|-1] >$;
        //pair of the last parameters
15:     if $m_l \notin M$ then
16:         $M \leftarrow M + m_l$;
17:     end if
18:     for $i = 0$ to $|M| - 2$ do
19:         $p_1^{SH_1} \leftarrow M[i][0]$; //first parameter of $SH_1$
20:         $p_{v_1}^{SH_1} \leftarrow M[i+1][0]$; //last parameter of $SH_1$
21:         $p_1^{SH_2} \leftarrow M[i][1]$; //first parameter of $SH_2$
22:         $p_{v_2}^{SH_2} \leftarrow M[i+1][1]$; //last parameter of $SH_2$
23:         if $Param^{SH_1} \subseteq Param^{SH_2}$ then
24:             $SH' \leftarrow \{SH_2\}$;
25:         else if $Param^{SH_2} \subseteq Param^{SH_1}$ then
```

```
26:           $SH' \leftarrow \{SH_1\}$;
27:       else if $FD_{SH_1\_SH_2} \neq \emptyset$ then
28:           for each $Param' \in MergeParameters(FD_{SH_1\_SH_2})$ do
29:               $Param^{SH_a} \leftarrow Param'; SH' \leftarrow SH' + SH_a$;
30:           end for
31:       else
32:           $SH' \leftarrow \{SH_1, SH_2\}$;
33:       end if
34:       $H' \leftarrow \{H'_a.extend(SH'_b) | (H'_a \in H') \wedge (SH'_b \in SH')\}$;
35:   end for
36: end if
37: if $id^{D_1} \simeq id^{D_2}$ then
38:   $H' \leftarrow H' \cup \{H_1, H_2\}$;
39:   return $H'$
40: else
41:   $p_1^{SH_{1'}} \leftarrow p_1^{H_1}$; //first parameter of $SH_{1'}$
42:   $p_{v'_1}^{SH_{1'}} \leftarrow M[0][0]$; //last parameter of $SH_{1'}$
43:   $p_1^{SH_{2'}} \leftarrow p_1^{H_2}$; //first parameter of $SH_{2'}$
44:   $p_{v'_2}^{SH_{2'}} \leftarrow M[0][1]$; //last parameter of $SH_{2'}$
45:   for each $H'_c \in H'$ do
46:       $H^{1'} \leftarrow SH_{1'}.extend(H'_c); H^{2'} \leftarrow SH_{2'}.extend(H'_c)$;
47:   end for
48:   $H^{1'} \leftarrow H^{1'} \cup \{H_1\}; H^{1'} \leftarrow H^{2'} \cup \{H_2\}$;
49:   return $H^{1'}, H^{2'}$
50: end if
```

4.1.1 Record of the matched parameters. The first step of the algorithm consists in matching the parameters of the two hierarchies and record the matched parameter pairs(L_1-L_9). If there is no matched parameter between the two hierarchies, the merging process stops (L_{11}-L_{12}).

4.1.2 Generation of the sub-hierarchy pairs. Then the algorithm generates pairs containing 2 sub-hierarchies (SH_1 and SH_2) of the original hierarchies whose lowest and highest level parameters are adjacent in the list of matched parameter pairs that we created in the previous step (L_{18}-L_{22}). To make sure that the last parameters of the two hierarchies are included in the sub-hierarchies, we also add the pair of the last parameters into the matched parameter pair (L_{14}-L_{17}).

Example 4.1. In Figure 2, for (a), we have $H1.Code \simeq H2.Code$, $H1.Department \simeq H2.Department$, $H1.Continent \simeq H2.Continent$. So for the first sub-hierarchy pair, the first parameter of SH_1 and SH_2 is *Code* and their last parameter is *Department*, so we have: $Param^{SH_1} =< Code, Department >$, $Param^{SH_2} =< Code, City, Department >$. In the second sub-hierarchy pair, we get the sub-hierarchy of H_1 from *Department* to *Continent* : $Param^{SH_1} =< Department, Region, Continent >$, and the sub-hierarchy of H_2 from *Department* to *Continent* : $Param^{SH_2} =< Department, Country, Continent >$. If the last parameters of the two original hierarchies do not match, like *Continent* of H_1 and *Country* of H_3 in (b), $< Continent, Country >$ is added into the matched parameter pair M of the algorithm so that the last sub-hierarchies of H_1 and H_3 are $Param^{SH_1} =< Department, Region, Continent >$ and $Param^{SH_3} =< Department, Country >$.

4.1.3 Merging of the sub-hierarchies. We then merge each sub-hierarchy pair to get a set of merged sub-hierarchies (SH') and

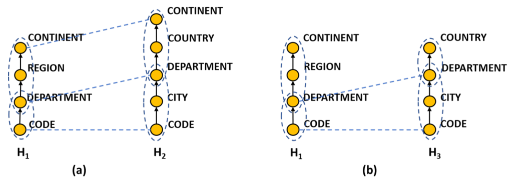


Figure 2: Example of generation of the sub-hierarchy pairs

combine each of these sub-hierarchy sets to get a set of merged hierarchies (H') (L_{23}-L_{35}).

The matched parameters will be merged into one parameter, so it's the unmatched parameters that we should deal with. We have 2 cases in terms of the unmatched parameters.

If one of the sub-hierarchies has no unmatched parameter, we obtain a sub-hierarchy set containing one sub-hierarchy whose parameter set is the same as the other sub-hierarchy (L_{23}-L_{26}).

Example 4.2. For the first parameter pair $SH_1 =< Code, Department >$ and $SH_2 =< Code, City, Department >$ of H_1 and H_2 in Figure 4. We see that SH_1 does not have any unmatched parameter, so the obtained sub-hierarchy set contains one sub-hierarchy whose parameter set is the same as $SH2$ which is $Param^{SH'} =<< Code, City, Department >>$.

The second case is that both two sub-hierarchies have unmatched parameters (L_{27}-L_{30}). We then see if these unmatched parameters can be merged into one or several hierarchies and discover their partial orders. Our solution is based on the functional dependencies (FDs) of these parameters. To be able to detect the FDs of the parameters of the two sub-hierarchies, we should make sure that there are intersections between the instances of these two sub-hierarchies which means that they should have same values on the root parameter of the sub-hierarchies. We keep only the FDs which have a single parameter in both hands and which can not be inferred by transitivity. These FDs are represented in the form of ordered set ($FD_{SH_1\_SH_2}$) are then treated by algorithm 2 *MergeParameters* to get the parameter sets of the merged sub-hierarchies. If it's not possible to discover the FDs, the two sub-hierarchies are impossible to be merged (L_{31}-L_{32}).

Algorithm 2 *MergeParameters* constructs recursively the parameter sets from the FDs in the form of ordered sets. In each recursion loop, for each one of these sets, we search for the other ones whose non-last (or non-first) elements have the same values and order as its non-first (or non-last) elements and then merge them (L_6-L_{21}). The recursion is finished until there are no more two sets being able to be merged (L_{22}-L_{31}).

Example 4.3. If we have $FD =<< A, B >, < B, C >, < B, F >, < C, E >, < D, B >>$. Like illustrated in Figure 3, in the first recursion, by merging the ordered set, we get $Param =<< A, B, C >, < A, B, F >, < B, C, E >, < D, B, C >, < D, B, F >>$, all the ordered sets in *FD* are merged, so there are only merged ordered set in *Param*. *Param* is then inputted to the second recursion, we then get the next $Param =<< A, B, C, E >, < D, B, C, E >>$ after the merging of the ordered sets, since $< A, B, F >$ and $< D, B, F >$ are not merged, they are also added into *Param*, and we get $Param =<< A, B, C, E >, < D, B, C, E >, < A, B, F >, < D, B, F >>$. In the final recursion, it's no more possible to merge any two ordered sets, so the parameter set of the final result of the hierarchy set is $<< A, B, C, E >, < D, B, C, E >, < A, B, F >, < D, B, F >>$.

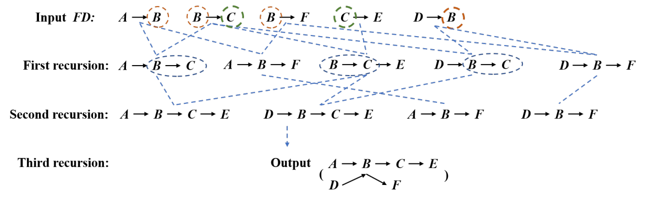


Figure 3: Example of parameter merging based on FDs

```
Algorithm 2 $MergeParameters(FD)$
Output: A set of parameter sets $Param$
 1: $l \leftarrow |FD|$;
 2: for $n \leftarrow 0$ to $l-1$ do
 3:     $fdmerged[n] \leftarrow False$; //Boolean indicating whether an element in $FD$ is mergerd
 4: end for
 5: $existmerged \leftarrow False$; //Boolean indicating whether there are elements that are merged in a recursion loop
 6: for $i \leftarrow 0$ to $l-1$ do
 7:     for $j \leftarrow i+1$ to $l$ do
 8:         if $FD[i][1:l-1] = FD[j][0:l-2]$ //$FD[a][b:c]$ represents the ordered set having the values and order from the $b$th element to the $c$th element of $FD[a]$ then
 9:             $Param^t \leftarrow FD[i][1:l-1]$ ;
10:             $Param^t \leftarrow Param^t + FD[j][l-1]$;
11:             $Param \leftarrow Param + Param^t$;
12:             $fdmerged[i], fdmerged[j], existmerged \leftarrow True$;
13:         end if
14:         if $FD[i][0:l-2] = FD[j][1:l-1]$ then
15:             $Param^t \leftarrow FD[j][1:l-1]$ ;
16:             $Param^t \leftarrow Param^t + FD[i][l-1]$;
17:             $Param \leftarrow Param + Param^t$;
18:             $fdmerged[i], fdmerged[j], existmerged \leftarrow True$;
19:         end if
20:     end for
21: end for
22: if $existmerged = True$ then
23:     for $m \leftarrow 0$ to $l-1$ do
24:         if $fdmerged[m] = False$ then
25:             $Param \leftarrow Param + FD[m]$;
26:         end if
27:     end for
28:     $Param \leftarrow MergeParameters(Param)$;
29: else
30:     $Param \leftarrow FD$;
31: end if
32: return $Param$
```

After the merging of each sub-hierarchy pair, we extend the final merged hierarchy set by the new merging result (L_{34}).

4.1.4 *Generation of the final hierarchy set.*

L_{37}-L_{49} concerns the generation of the final hierarchy set. The two original hierarchies may have different instances, so there may be empty values in the instances of the merged hierarchies. Some empty values can be completed, which is introduced in the next section of dimension merging. But not all empty values can be completed. The empty values generate the incomplete hierarchies and make the analysis difficult. Inspired by the concept of the structural repair[1], we also add the two original hierarchies into the final hierarchy set. Then for a parameter which appears in different hierarchies, it can be divided into different parameters in different hierarchies of the hierarchy set so that each hierarchy is complete. Thus, for the multidimensional schema that we get, we provide an analysis form like shown in Figure 4. In the analysis form, one parameter can be marked with different numbers if it is in different hierarchies.

For the generation of the final hierarchy set, we discuss 2 cases where the 2 hierarchies have the matched root parameters which means their dimensions are the same analysis axis and the opposite case which will lead to 2 kinds of output results (one or two sets of merged hierarchies).

If the root parameters of the two original hierarchies match, we simply add the two original hierarchies into the merged hierarchy set obtained in the previous step to get one final merged hierarchy set. (L_{37}-L_{39}).

Example 4.4. For the hierarchies H_1 and H_2 in Figure 4, we combine the merged hierarchy obtained in *Example* 4.4 with the result gained in *Example* 4.2 to get the merged hierarchy H_m : $< Code, City, Department, Region, Country, Continent >$. We add H_m into the hierarchy set H' and then also add the original hierarchies H_1 and H_2. Thus H' is the final merged hierarchy set.

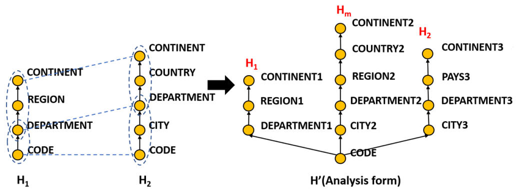


Figure 4: Hierarchy merging example

If the root parameters of the two original hierarchies do not match, we will get two merged hierarchy sets instead of one. For each original hierarchy, the final merged hierarchy set will be the extension of the sub-hierarchy containing all the parameters which are not included in any one of the sub-hierarchies created before ($SH_{1'}$ and $SH_{2'}$) with the merged hierarchy set that we get plus this original hierarchy itself (L_{41}-L_{49}).

Example 4.5. In Figure 5, between H_1 and H_3, we have $H_1.Department \simeq H_3.Department$ and $H_1.Continent \simeq H_3.Continent$. We can then get one sub-hierarchy pair in which there are 2 sub-hierarchies containing parameter sets $< Department, Region, Continent >$ and $< Department, Country, Continent >$. By merging the sub-hierarchy pairs, we get the merged hierarchy whose parameter set is $< Department, Region, Country, Continent >$. For H_1, the remaining part $< Code >$ is associated to it to get the merged hierarchy H^1_{13}. We then get the merged hierarchy set of H_1 containing H_1 and H^1_{13}. We do the same thing for H_3 and get the merged hierarchy set containing H_3 and H^2_{13}.

4.2 Dimension merging

This section concerns the merging of two dimensions having matched attributes which is realized by algorithm 3 *MergeDimensions*. We consider both the schema and instance levels for the merging of dimensions. The schema merging is based on the merging of hierarchies. Concerning the instances, we have 2 tasks: merging the instances and completing the empty values.

Algorithm 3 $MergeDimensions(D_1, D_2)$

Output: One merged dimension D' or two merged dimensions $D^{1'}$ and $D^{2'}$

```
1: if $id^{D_1} \simeq id^{D_2}$ then
2:     $H^{D'} \leftarrow \emptyset$;
3:     for each $H_i^{D_1} \in H^{D_1}$ do
4:         for each $H_j^{D_2} \in H^{D_2}$ do
5:             $H^{D'} \leftarrow H^{D'} \cup MergeHierarchies(H_i^{D_1}, H_j^{D_2})$;
6:         end for
7:     end for
8:     $A^{D'} \leftarrow A^{D_1} \cup A^{D_2}; H^m \leftarrow H^{D'} \setminus (H^{D_1} \cup H^{D_2})$;
9:     $CompleteEmpty(D', D', H^m)$;
10:    return $D'$
11: else
12:    $H^{D^{1'}}, H^{D^{2'}}, A^{D^{1'}}, A^{D^{2'}} \leftarrow \emptyset$;
13:    for each $H_i^{D_1} \in H^{D_1}$ do
14:        for each $H_j^{D_2} \in H^{D_2}$ do
15:            $H^{1'}, H^{2'} \leftarrow MergeHierarchies(H_i^{D_1}, H_j^{D_2})$;
16:            $H^{D^{1'}} \leftarrow H^{D^{1'}} \cup H^{1'}; H^{D^{2'}} \leftarrow H^{D^{2'}} \cup H^{2'}$;
17:        end for
18:    end for
19:    for each $H_u^{D^{1'}} \in H^{D^{1'}}$ do
20:        $A^{D^{1'}} \leftarrow A^{D^{1'}} \cup Param^{H_u^{D^{1'}}}$;
21:    end for
22:    for each $H_v^{D^{2'}} \in H^{D^{2'}}$ do
23:        $A^{D^{2'}} \leftarrow A^{D^{2'}} \cup Param^{H_v^{D^{2'}}}$;
24:    end for
25:    $H^{m_1} \leftarrow H^{D'} \setminus H^{D_1}; H^{m_2} \leftarrow H^{D'} \setminus H^{D_2}$;
26:    $CompleteEmpty(D^{1'}, D^{2'}, H^{m_1})$;
27:    $CompleteEmpty(D^{2'}, D^{1'}, H^{m_2})$;
28:    return $D^{1'}, D^{2'}$
29: end if
```

4.2.1 Schema merging. If the root parameters of the two dimensions match, the algorithm generates a merged dimension (L_1-L_8). The hierarchy set of the merged dimension is the union of the hierarchy sets generated by merging every 2 hierarchies of the original dimensions (L_3-L_7). We also get a hierarchy set containing only the merged hierarchies but no original hierarchies (H^m) which is to be used for the complement of the empty values (L_8). The attribute set of the merged dimension is the union of the attribute sets of the original dimensions (L_8).

Example 4.6. Given 2 original dimensions D_1 and D_2 in Figure 8 and their instances in Figure 6, we can get the merged dimension schema D' in Figure 8. In D', H_1 and H_2 are the original hierarchies of D_1, H_3 and H_4 are those of D_2, H_{13} is a merged hierarchy of H_1 and H_3, and H_{24} is a merged hierarchy of H_2 and H_4. We can thus get $H^{D'} = \{H_1, H_2, H_3, H_4, H_{13}, H_{24}\}$, $H_m = \{H_{13}, H_{24}\}$, $A^{D'} = \{Code, City, Department, Region, Country, Continent, Profession, Subcategory, Category\}$

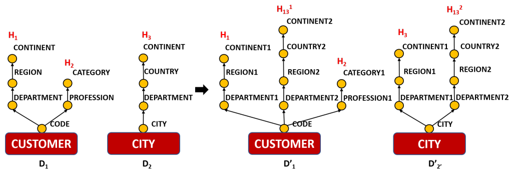


Figure 5: Dimension merging example (schema)

When the root parameters of the two dimensions don't match, we will get a merged dimension for each original dimension, which is realized by L_{13}-L_{25}. For each original dimension, the hierarchy set of its corresponding merged dimension is the union of all hierarchy sets generated by merging every 2 hierarchies of the original dimensions (L_{13}-L_{18}), the attribute set is the union of the attributes of each hierarchy in the merged dimension (L_{19}-L_{24}). Similar to the first case, we get a hierarchy set containing only the merged hierarchies for each original dimension (H^{m_1} and H^{m_2}) (L_{26}-L_{27}).

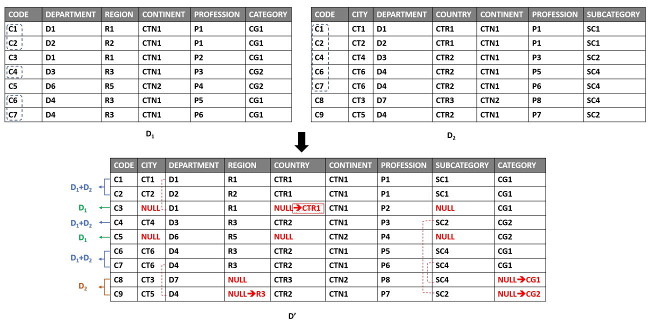
Example 4.7. Given 2 original dimensions D_1 and D_2 in Figure 5 and their instances in Figure 7, after the execution of algorithm 3 *MergeDimensions*, we can get the merged dimension schema $D^{1'}$ and $D^{2'}$ in Figure 5. In $D^{1'}$, H_1 and H_2 are the original hierarchies of D_1, H_{13}^1 is the merged hierarchy of H_1 and H_3. In $D^{2'}$, H_3 is the original hierarchy of D_2, H_{13}^2 is the merged hierarchy of H_1 and H_3. So for D_1, we have $H^{D^{1'}} = \{H_1, H_2, H_{13}^1\}$, $H_{m1} = \{H_{13}^1\}$, $A^{D^{1'}} = \{Code, Department, Region, Country, Continent, Profession, Category\}$, while for D_2, we get $H^{D^{2'}} = \{H_3, H_{13}^2\}$, $H_{m2} = \{H_{13}^2\}$, $A^{D^{2'}} = \{City, Department, Region, Country, Continent\}$

4.2.2 Instance merging and complement. When the root parameters of the two dimensions match, the instance of the merged dimension is obtained by the union of the two original dimension instances which means that we insert the data of the two original dimension tables into the merged dimension table and merge the lines which have the same root parameter instance (L_9).

Example 4.8. The instance merging result of Example 4.2.1 is presented in Figure 6. All the data in the original dimension tables D_1, D_2 are integrated into the merged dimension table D'. The original tables of the instances are marked on the left of the merged table D' with different colors. There are instances coming from both D_1 and D_2, which means that they have the same root parameter in D_1 and D_2, and are therefore merged together.

The attribute set of the merged dimension contains all the attributes of two original dimensions, while the original dimensions may contain their unique attributes. So there may be empty values in the merged dimension table on the instances coming from only one of the original dimension tables and we should complete the empty values on the basis of the existing data (L_9).

The complement of the empty values is realized by Algorithm *CompleteEmpty* where the input $D^{1'}$ is the merged dimension table having empty values to be completed, $D^{2'}$ is the merged dimension table which provides the completed values and H_m is the hierarchy



CODE	DEPARTMENT	REGION	CONTINENT	PROFESSION	CATEGORY
C1	D1	R1	CTN1	P1	CG1
C2	D2	R2	CTN1	P1	CG1
C3	D1	R1	CTN1	P2	CG1
C4	D3	R3	CTN1	P3	CG2
C5	D6	R5	CTN2	P4	CG2
C6	D4	R3	CTN1	P5	CG1
C7	D4	R3	CTN1	P6	CG1

D_1

CODE	CITY	DEPARTMENT	COUNTRY	CONTINENT	PROFESSION	SUBCATEGORY
C1	CT1	D1	CTR1	CTN1	P1	SC1
C2	CT2	D2	CTR1	CTN1	P1	SC1
C4	CT4	D3	CTR2	CTN1	P3	SC2
C6	CT6	D4	CTR2	CTN1	P5	SC4
C7	CT6	D4	CTR2	CTN1	P6	SC4
C8	CT3	D7	CTR3	CTN2	P8	SC4
C9	CT5	D4	CTR2	CTN1	P7	SC2

D_2

	CODE	CITY	DEPARTMENT	REGION	COUNTRY	CONTINENT	PROFESSION	SUBCATEGORY	CATEGORY
D_1+D_2	C1	CT1	D1	R1	CTR1	CTN1	P1	SC1	CG1
	C2	CT2	D2	R2	CTR1	CTN1	P1	SC1	CG1
D_1	C3	NULL	D1	R1	NULL→CTR1	CTN1	P2	NULL	CG1
D_1+D_2	C4	CT4	D3	R3	CTR2	CTN1	P3	SC2	CG2
D_1	C5	NULL	D6	R5	NULL	CTN2	P4	NULL	CG2
D_1+D_2	C6	CT6	D4	R3	CTR2	CTN1	P5	SC4	CG1
	C7	CT6	D4	R3	CTR2	CTN1	P6	SC4	CG1
D_2	C8	CT3	D7	NULL	CTR3	CTN2	P8	SC4	NULL→CG1
	C9	CT5	D4	NULL→R3	CTR2	CTN1	P7	SC2	NULL→CG2

D′

Figure 6: Dimension merging example (instance)

set of $D^{1'}$ containing only merged hierarchies but no original hierarchies. In this discussed case, D' is inputted as both $D^{1'}$ and $D^{2'}$ in *CompleteEmpty* since we get one merged dimension including all data of two original dimensions (L_{11}).

Algorithm 4 $CompleteEmpty(D^{1'}, D^{2'}, H^m)$

```
1:  for each $H_a^m \in H^m$ do
2:      $I^n \leftarrow \emptyset$;
3:      $I^n \leftarrow I^n \cup \{i_k^{D^{1'}} \in I^{D^{1'}} | (i_k^{D^{1'}}.p_1^{H_a^m}$ is not null$) \wedge (\exists p_v^{H_a^m} \in Param^{H_a^m}, i_k^{D^{1'}}.p_v^{H_a^m}$ is null$)\}$;
4:      for each $i_b^n \in I^n$ do
5:          $P^n \leftarrow \{p_v^{H_a^m} \in Param^{H_a^m} | i_b^n.p_v^{H_a^m}$ is null$\}$;
6:          $P^r \leftarrow \{p_v^{H_a^m} \in Param^{H_a^m} | (i_b^n.p_v^{H_a^m}$ is not null$) \wedge (\forall p_s^n \in P^n, p_v^{H_a^m} \preceq_H p_s^n)\}$;
7:          if $\exists i_u^{D^{2'}} \in I^{D^{2'}}\ \exists p_w^r \in P^r, (i_u^{D^{2'}}.p_w^r = i_b^n.p_w^r) \wedge (\forall p_q^n \in P^n, i_u^{D^{2'}}.p_q^n$ is not null$)$ then
8:              for each $p_c^n \in P^n$ do
9:                  $i_b^n.p_c^n \leftarrow i_u^{D^{2'}}.p_c^n$;
10:             end for
11:         end if
12:     end for
13: end for
```

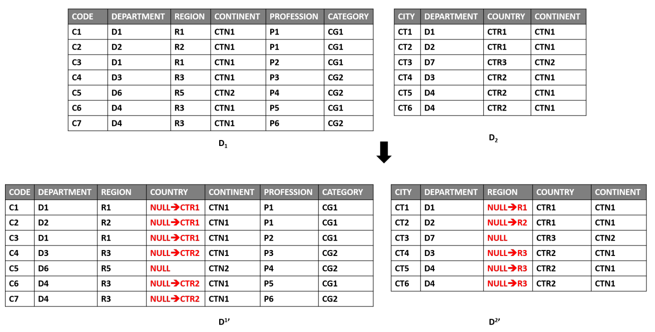
For an empty value, we search for an instance which has the same value as the instance of this empty value on one of the parameters rolling up to the parameter of the empty value and whose value of the parameter of the empty value is not empty, we can then fill the empty by this non-empty value. The complement of the empty values is also possibly a change of hierarchies. Nevertheless, after completing the empty values of an instance, there may be some completed parameters which are not included in the hierarchies of the instance, so the complement of such values does not make sense in this case. The possible change of the hierarchy is from the hierarchies containing less parameters to those containing more parameters. We know that the merged hierarchies contain more parameters than their corresponding original hierarchies. Hence, before the complement of an instance, we will first look at the merged hierarchies to decide which parameter values can be completed.

In algorithm 4 *CompleteEmpty* which aims to complete the empty values, for each hierarchy in the merged hierarchy set we see, if **(a)** there exists instances in the merged dimension table which contains empty values on the parameters of this hierarchy (L_3) and **(b)** where the value of the second lowest parameter is not empty (L_3). The condition a is basic because we need empty values to be completed. Since we will complete the empty values by the other lines of the merged dimension table, we can only complete the empty values based on the non-id parameters since the id is unique, so if the second lowest parameter is empty, it can never be completed so that the hierarchy can never be completed. That's why we have the condition b. For each one of the instances satisfying these conditions (I^n), we search for the parameters (P^n) having empty values (L_5) and to make sure that each one of them can be completed, we search also for the parameters (P^r) which roll up to the lowest of them and to which we refer to complete the empty values (L_6). We can then complete the empty values like discussed in the previous paragraph (L_7-L_{11}).

Example 4.9. After the merging in *Example* 4.9, we get the empty values of D' which are in red in Figure 6. The merged hierarchies are H_{13} and H_{24} as illustrated in Figure 5. For H_{13}, the instances of code $C3$ and $C5$ have empty values on the second root parameter *City*, which do not satisfy the condition b. As we can see, for the instance of $C3$, although the value of *Country* can be retrieved through the value of *Department* which is the same as the instance of $C1$, the value of *City* can not be completed and thus we should give up this complement. For the instance of $C9$, the value of *Region* is completed by $C7$ which has the same value of *Department* and whose value of *Region* is not empty. When it's the turn of H_{24}, values of *Category* of $C8$ and $C9$ are completed in the same way.

When the root parameters of the two dimensions don't match, the instance merging and complement are done by L_{26}-L_{27}. The values of the attributes of one of the dimension tables coming from the other dimension table are empty, so there is only instance complement but no merging. We also call algorithm 4 *CompleteEmpty* to complete the instances for each one of the merged dimension tables.

Example 4.10. The instance merging and complement of the example for *Example* 4.8 is demonstrated in Figure 7. For $D^{1'}$, *Country* comes from the dimension table D_2, so the values of *Country* are completed by the values in $D^{2'}$. The same operation is also done for *Region* of $D^{2'}$.



CODE	DEPARTMENT	REGION	CONTINENT	PROFESSION	CATEGORY
C1	D1	R1	CTN1	P1	CG1
C2	D2	R2	CTN1	P1	CG1
C3	D1	R1	CTN1	P2	CG1
C4	D3	R3	CTN1	P3	CG2
C5	D6	R5	CTN2	P4	CG2
C6	D4	R3	CTN1	P5	CG1
C7	D4	R3	CTN1	P6	CG2

D_1

CITY	DEPARTMENT	COUNTRY	CONTINENT
CT1	D1	CTR1	CTN1
CT2	D2	CTR1	CTN1
CT3	D7	CTR3	CTN2
CT4	D3	CTR2	CTN1
CT5	D4	CTR2	CTN1
CT6	D4	CTR2	CTN1

D_2

CODE	DEPARTMENT	REGION	COUNTRY	CONTINENT	PROFESSION	CATEGORY
C1	D1	R1	NULL→CTR1	CTN1	P1	CG1
C2	D2	R2	NULL→CTR1	CTN1	P1	CG1
C3	D1	R1	NULL→CTR1	CTN1	P2	CG1
C4	D3	R3	NULL→CTR2	CTN1	P3	CG2
C5	D6	R5	NULL	CTN2	P4	CG2
C6	D4	R3	NULL→CTR2	CTN1	P5	CG1
C7	D4	R3	NULL→CTR2	CTN1	P6	CG2

$D^{1'}$

CITY	DEPARTMENT	REGION	COUNTRY	CONTINENT
CT1	D1	NULL→R1	CTR1	CTN1
CT2	D2	NULL→R2	CTR1	CTN1
CT3	D7	NULL	CTR3	CTN2
CT4	D3	NULL→R3	CTR2	CTN1
CT5	D4	NULL→R3	CTR2	CTN1
CT6	D4	NULL→R3	CTR2	CTN1

$D^{2'}$

Figure 7: Dimension merging example (instance)

4.3 Star merging

In this section, we discuss the merging of two stars. Having two stars, we can get a star schema or a constellation schema because the fact table of each schema may be merged into one schema or not. The star merging is related to the dimension merging and fact merging. Two stars are possible to be merged only if there are dimensions having matched root parameters between them.

Algorithm 5 *MergeAllDimensions*(S_1, S_2)

```
Output: A set of merged dimensions $D^{S'}$
1:  for each $D_i^{S_1} \in D^{S_1}$ do
2:      for each $D_j^{S_2} \in D^{S_2}$ do
3:          if $id^{D_i^{S_1}} \not\simeq id^{D_j^{S_2}}$ then
4:              $D_i^{S_1}, D_j^{S_2} \leftarrow MergeDimensions(D_i^{S_1}, D_j^{S_2})$;
5:          end if
6:      end for
7:  end for
8:  $D^{S'} \leftarrow \emptyset$;
9:  for each $D_u^{S_1} \in D^{S_1}$ do
10:     for each $D_v^{S_2} \in D^{S_2}$ do
11:         if $id^{D_u^{S_1}} \simeq id^{D_v^{S_2}}$ then
12:             $D^{S'} \leftarrow D^{S'} \cup MergeDimensions(D_u^{S_1}, D_v^{S_2})$;
13:         end if
14:     end for
15: end for
16: for each $D_k^{S'} \in D^{S'}$ do
17:     for each $H_m^{D_k^{S'}} \in H^{D_k^{S'}}$ do
18:         if $\nexists i_r^{D_k^{S'}} \in I^{D_k^{S'}}, (i_r^{D_k^{S'}}$ is on $H_m^{D_k^{S'}}) \vee (i_r^{D_k^{S'}}$ is only on
            $H_m^{D_k^{S'}} \wedge (H_m^{D_k^{S'}} \in H^{D^{S_1}} \vee H_m^{D_k^{S'}} \in H^{D^{S_2}}))$ then
19:             $H^{D_k^{S'}} \leftarrow H^{D_k^{S'}} - H_m^{D_k^{S'}}$;
20:         end if
21:     end for
22: end for
23: return $D^{S'}$
```

For the dimensions of the two stars, we have two cases: 1. The two stars have the same number of dimensions and for each dimension of one schema, there is a dimension having matched root parameters in the other schema. 2. There exists at least one dimension between the two stars which does not have a dimension having a matched root parameter in the other.

The dimension merging of two stars is common for the two cases which is done by algorithm 5 *MergeAllDimension*. We first merge every two dimensions of the two stars which have unmatched root parameters because the merging of such dimensions is able to complete the original dimensions with complementary attributes (L_1-L_7). Then the dimensions having matched root parameters are merged to generate the merged dimensions of the merged multidimensional schema (L_8-L_{15}). After the merging and complement of the instances of the dimension tables, there may be some merged hierarchies to which none of the instances belong. In this case, if there will be no more update of the data, such hierarchies should be deleted. There may also be original hierarchies in the merged dimensions such that there is no instance which belongs to them but does not belong to any merged hierarchy containing all the parameters of this original hierarchy. The instances belonging to this kind of hierarchies belong also to other hierarchies which contains more parameters,so they become useless and should also be deleted (L_{18}-L_{19}).

Example 4.11. For the merging of the dimensions of two stars S_1 and S_2 in Figure 8. The dimension *Product* of S_1 and the dimension *Customer* of S_2 are firstly merged since their root parameters don't match but they have other matched parameters. There are then attributes of dimension *Customer* of S_2 added into dimension *Product* of S_1. The two dimensions *Customer* and the two dimensions *product* have matched root parameters, so they are merged into the final star schema. After the merging and complement of the instance, we verify each hierarchy in the merged dimension tables. If the merging of S_1.*Customer* and S_2.*Customer* is as shown in Figure 5 at the schema level and in Figure 6 at the instance level. In their merged dimension table D'. We can find that all the instances belonging to H_4 also belong to H_{24} which is a merged hierarchy containing all the parameters of H_4, so H_4 should be deleted.

We then discuss the merging of the other elements in the two cases which is processed by algorithm 6 *MergeStar*:

Algorithm 6 *MergeStar*(S_1, S_2)

```
Output: A merged multidimensional schema which may be a star schema
$S'$ or a merged constellation schema $C'$
1: if $(|D^{S_1}| = |D^{S_1}|) \wedge (\forall D_i^{S_1} \in D^{S_1} \exists D_j^{S_2} \in D^{S_2}, id^{D_i^{S_1}} \simeq id^{D_j^{S_2}})$
   then
2:     $D^{S'} \leftarrow MergeAllDimensions(S_1, S_2)$;
3:     $M^{F^{S'}} \leftarrow M^{F^{S_1}} \cup M^{F^{S_2}}$; $I^{F^{S'}} \leftarrow I^{F^{S_1}} \cup I^{F^{S_2}}$;
4:     $IStar^{F^{S'}} \leftarrow IStar^{F^{S_1}} \cup IStar^{F^{S_2}}$;
5:     return $S'$
6: else
7:     $D^{S'} \leftarrow MergeAllDimensions(S_1, S_2)$; $F^{C'} \leftarrow \{F_1^{S'}, F_2^{S'}\}$;
8:     return $C'$
9: end if
```

For the first case, we merge the two fact tables into one fact table and get a star schema. The measure set of the merged star schema is the union of the 2 original measures (L_3). The fact instances are the union of the measure instances of the two input star schemata (L_4). The function associating fact instances to their linked dimension instances of the merged schema is also the union of the functions of the original schemata (L_4).

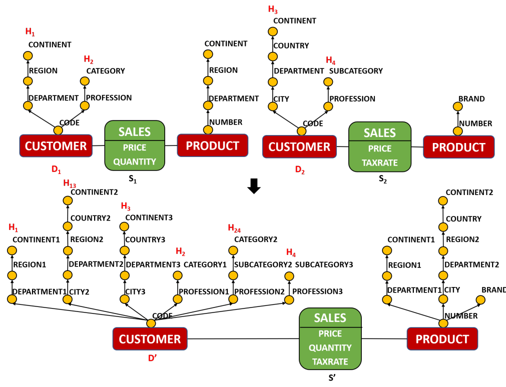
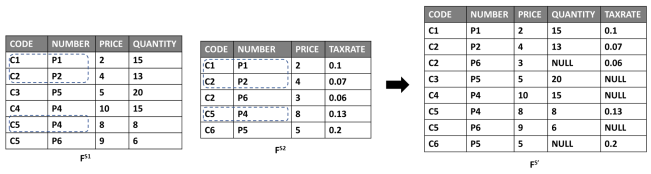


Figure 8: Star merging example (schema)



CODE	NUMBER	PRICE	QUANTITY
C1	P1	2	15
C2	P2	4	13
C3	P5	5	20
C4	P4	10	15
C5	P4	8	8
C5	P6	9	6

F^{S1}

CODE	NUMBER	PRICE	TAXRATE
C1	P1	2	0.1
C2	P2	4	0.07
C2	P6	3	0.06
C5	P4	8	0.13
C6	P5	5	0.2

F^{S2}

CODE	NUMBER	PRICE	QUANTITY	TAXRATE
C1	P1	2	15	0.1
C2	P2	4	13	0.07
C2	P6	3	NULL	0.06
C3	P5	5	20	NULL
C4	P4	10	15	NULL
C5	P4	8	8	0.13
C5	P6	9	6	NULL
C6	P5	5	NULL	0.2

$F^{S'}$

Figure 9: Star merging example (instance)

Example 4.12. For the two original star schemata in Figure 8, the dimension merging is discussed above so we mainly focus on the merging of fact table instances here. The dimensions *Customer*, *Product* of S_1 have respectively matched root parameters in the dimensions *Customer*, *Product* of S_2. They also have the same number of dimensions. Therefore we get a merged star schema S', the original fact tables are merged by merging the measures of S_1 and S_2 to get the fact table of S'. At the instance level, in Figure 9, we have the instances of the fact tables, for the instances of F^{S_1} and F^{S_2}, the framed parts are the instances having the common linked dimension instances, so they are merged into the merged fact table $F^{S'}$, the other instances are also integrated in $F^{S'}$ but with empty values in the merged instances, but they will not have big impacts on the analysis, so they will not be treated particularly.

For the second case, since there are unmatched dimensions, the merged schema should be a constellation schema. The facts of the original schemata have no change at both the schema and instance levels and compose the final constellation. (L_8)

Example 4.13. This example is simplified in Figure 10 due to the space limit. For the original star schemata S_1 and S_2, they have dimensions *Customer* which have the matched root parameters. They also have their unique dimensions: *Time* of S_1 and *Product* of S_2. So the merged schema is a constellation schema generated by merging the dimensions *Customer* and by keeping the other dimensions and fact tables. At the instance level, we just have a new merged dimension table of *Customer*, the other dimension and fact tables remain unchanged.

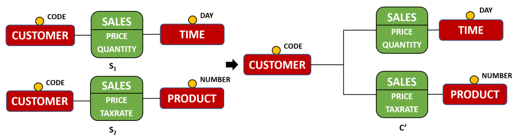


Figure 10: Star merging example (schema)

5 EXPERIMENTAL ASSESSMENTS

To validate the effectiveness of our approach, we applied our algorithms on benchmark data. Unfortunately, we did not find a suitable benchmark for our problem. So, we adapted the datasets of the TPC-H benchmark to generate different DWs. Originally, the TPC-H benchmark serves for benchmarking decision support systems by examining the execution of queries on large volumes of data. Because of space limit, we put the test results in github[1].

[1]https://github.com/Implementation111/Multidimensional-DW-merging

5.1 Technical environment and Datasets

The algorithms were implemented by Python 3.7 and were executed on a processor of Intel(R) Core(TM) i5-8265U CPU@ 1.60GHz with a 16G RAM. The data are implemented in R-OLAP format through the Oracle 11g DBMS. The TPC-H benchmark provides a pre-defined relational schema[2] with 8 tables and a generator of massive data.

First, we generated 100M of data files, there are respectively 600572, 15000, 25, 150000, 20000, 80000, 5, 1000 tuples in the table of *Lineitem*, *Customer*, *Nation*, *Orders*, *Part*, *Partsupp*, *Region* and *Supplier*. Second, to have more deeper hierarchies, we included the data of *Nation* and *Region* into *Customer* and *Supplier*, and those of *Partsupp* into *Part*. Third, we transformed these files to generate two use cases by creating 2 DWs for each case. To make sure that there are both common and different instances in different DWs, for each dimension, instead of selecting all the corresponding data, we selected randomly 3/4 of them. For the fact table, we selected the measures related to these dimension data. Since the methods in the related work do not have exactly the same treated components or objective as the ours, we do not have comparable baseline in our experiments.

5.2 Star schema generation

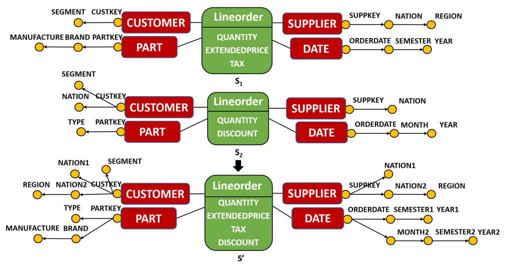


Figure 11: Star schema generation

The objective of this experiment is to merge two star schemata having the same 4 dimensions with the matched lowest level of granularity for each dimension.

After executing our algorithms, we obtain one star schema as shown in Figure 11 which is consistent with the expectations. The parameters of the hierarchies satisfy the relationships of functional dependency. The run time is 30.70s. The 3 dimensions *Supplier*, *Part*, *Date* of the original DWs are merged. Between the different dimensions S_1.*Supplier* and S_2.*Customer*, there is a matched attribute *Nation*, so they are also merged such that S_1.*Supplier* provides S_2.*Customer* with the attribute *Region*. Then the *Customer* in the merged DW also has the attribute *Region*. We can also observe that normally, in the merged schema, there should be the original hierarchy $Orderdate \longrightarrow Month \longrightarrow Year$ of S_2.*Date* but which is deleted. By looking up in the table, we find that there is no tuple which belongs to this hierarchy but not to $Orderdate \longrightarrow Month \longrightarrow Semester \longrightarrow year$, that's why it is removed.

At the instance level, the result is shown in github. Table 1 shows the number of tuples of the original DWs (N_1, N_2), of the merged DW (N') and the number of the common tuples ($N_\cap$) (tuples having

[2]http://tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.18.0.pdf

	Customer	Supplier	Part	Orderdate	Lineorder
N_1	11250	750	15000	1804	252689
N_2	11250	750	15000	1804	252821
$N_\cap$	8439	556	11261	1349	105345
N'	14061	944	18739	2259	400165

Table 1: Number of tuples

	Customer.Region	Supplier.Region	Orderdate.Semester
N_1	X	X	1804
N_2	X	750	X
N'	9713	846	2259
N_+	9713	96	455

Table 2: Number of attributes

the same dimension key in the original DWs). For each dimension or fact table, $N' = N_1 + N_2 - N_\cap$, we can thus confirm that there is no addition or loss of data. For each tuple in the original tables, we verify that the all the values are the same with the values in the merged table. We also find that there are some empty values of the attribute *Region* in the dimension *Customer* and *Supplier* and the attribute *Semester* of the dimension *Orderdate* which are completed. Table 2 shows the number of these attributes in the original DWs (N_1, N_2) and in the merged DW (N'), we can then get the number of the completed values N_+ for these attributes. They meet the relationship $N' = N_1 + N_2 + N_+$.

5.3 Constellation schema generation

The objective of this experiment is to merge two star schemata having the same 2 dimensions (*Customer*, *Supplier*) with the same lowest level of granularity for each dimension, as well as 2 different dimensions (S_1.*Part* and *S*2.*Date*).

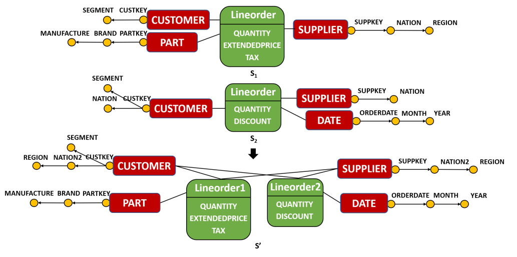


Figure 12: Constellation schema generation

At the schema level, the second test generates a constellation schema like shown in Figure 12. The run time is 32.13s. As expected, the 2 dimensions *Customer*, *Supplier* of the original DWs are merged, the other dimension and fact tables are not merged. The dimension *Customer* gains a new attribute *Region* by the merging between S_1.*Supplier* and S_2.*Customer*. We can see that the hierarchy $Custkey \longrightarrow nation$ of *Customer* which should be in the merged schema is deleted because there is no tuple which belongs to this hierarchy but not to $Custkey \longrightarrow nation \longrightarrow Region$. The hierarchy $Suppkey \longrightarrow nation$ of *Supplier* is removed due to the same reason.

At the instance level, the data of experiment can be found in github. They also meet $N' = N_1 + N_2 - N_\cap$. There are empty values of the attribute *Region* in the dimension *Customer* and *Supplier* which are completed which meet $N' = N_1 + N_2 + N_+$.

We got the results conforming to our expectations in the tests, we can thus conclude that our algorithms work well for the different cases discussed at both schema and instance levels.

6 CONCLUSION AND FUTURE WORK

In this paper, we define an automatic approach to merge two different star schema-modeled DWs, by merging multidimensional schema elements including hierarchies, dimensions and facts at the schema and instance levels. We define the corresponding algorithms, which consider different cases. Our algorithms are implemented and illustrated by various examples.

Since we only discuss the merging of DWs modeled as star schemata in this paper, which is only one (albeit common) possible DW design, we plan to extend our approach by adding the merging of DWs modelled as constellation schemata in the future. There may also be so-called weak attributes in DW components. Thus, we will consider them in future work. Our goal is to provide a complete approach that is integrated in our previous work concerning the automatic integration of tabular data in DWs.

ACKNOWLEDGMENTS

The research depicted in this paper is funded by the French National Research Agency (ANR), project ANR-19-CE23-0005 BI4people (Business Intelligence for the people).

REFERENCES

[1] Sina Ariyan and Leopoldo Bertossi. 2011. Structural Repairs of Multidimensional Databases. In *Inter. Workshop on Foundations of Data Management*, Vol. 748.
[2] Marko Banek, Boris Vrdoljak, A. Min Tjoa, and Zoran Skočir. 2007. Automating the Schema Matching Process for Heterogeneous Data Warehouses. In *Data Warehousing and Knowledge Discovery*. 45–54.
[3] S. Bergamaschi, M. Olaru, S. Sorrentino, and M. Vincini. 2011. Semi-automatic Discovery of Mappings Between Heterogeneous Data Warehouse Dimensions. *J. of Computing and Information Technology* (dec 2011), 38–46.
[4] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proc. VLDB Endow.* 4, 11 (aug 2011), 695–701.
[5] Elhaj Elamin, Amer Alzaidi, and Jamel Feki. 2018. A Semantic Resource Based Approach for Star Schemas Matching. *IJDMS* 10, 6 (dec 2018).
[6] S. Anitha Elavarasi, J. Akilandeswari, and K. Menaga. 2014. A Survey on Semantic Similarity Measure. *Inter. J. of Research in Advent Technology* 2 (mar 2014).
[7] Jamel Feki, Jihen Majdoubi, and Faïez Gargouri. 2005. A Two-Phase Approach for Multidimensional Schemes Integration. In *17th Inter. Conference on Software Engineering and Knowledge Engineering*. 498–503.
[8] M. Kwakye, I. Kiringa, and H. L. Viktor. 2013. Merging Multidimensional Data Models: A Practical Approach for Schema and Data Instances. In *5th Inter. Conference on Advances in Databases, Data, and Knowledge Applications.*
[9] Salvatore T. March and Alan R. Hevner. 2007. Integrated decision support systems: A data warehousing perspective. *Decis. Support Syst.* 43, 3 (apr 2007), 1031 – 1043.
[10] Lingling Meng, Runqing Huang, and Junzhong Gu. 2013. A review of semantic similarity measures in wordnet. *IJHIT* 6 (jan 2013).
[11] Marius-Octavian Olaru and Maurizio Vincini. 2012. A Dimension Integration Method for a Heterogeneous Data Warehouse Environment. In *Inter. Conf. on Data Communication Networking, e-Business and Optical Communication Systems.*
[12] Christoph Quix, David Kensche, and Xiang Li. 2007. Generic Schema Merging. In *Advanced Information Systems Engineering*. 127–141.
[13] Franck Ravat, Olivier Teste, Ronan Tournier, and Gilles Zurfluh. 2008. Algebraic and Graphic Languages for OLAP Manipulations. *Inter. J. of Data Warehousing and Mining* 4 (jan 2008), 17–46.
[14] Oscar Romero and Alberto Abelló. 2009. A Survey of Multidimensional Modeling Methodologies. *Inter. J. of Data Warehousing and Mining* 5, 2 (apr 2009).
[15] Riccardo Torlone. 2008. Two approaches to the integration of heterogeneous data warehouses. *Distributed and Parallel Databases* 23 (feb 2008), 69–97.