Efficient Algorithms for Repairing Inconsistent Dimensions in Data Warehouses

Raúl Arredondo
Universidad del Bío-Bío
Concepción, Chile
rarredon@ubiobio.cl

Mónica Caniupán
Universidad del Bío-Bío
Concepción, Chile
mcaniupa@ubiobio.cl

***Abstract*—Dimensions in Data Warehouses (DWs) are usually modeled as a hierarchical set of categories called the *dimension schema*. To guarantee summarizability, this is, the capability of using pre-computed answers at lower levels to compute answers at higher levels, a dimension is required to be *strict* and *covering*, meaning that every element of the dimension must be connected to a unique ancestor in each of its ancestor categories. In practice, rollup relations of dimensions need to be reclassified to correct errors or to adapt the data to changes. After these operations the dimension may become non-strict. A *minimal r-repair* is a new dimension that is strict and covering, is obtained from the original dimension through a minimum number of changes, and keeps the set of reclassifications. In the general case finding an *r-repair* for a dimension is NP-complete. We present efficient polynomial time algorithms to compute a single *r-repair* for dimensions that contain one conflicting level and become inconsistent after one reclassification of elements.**

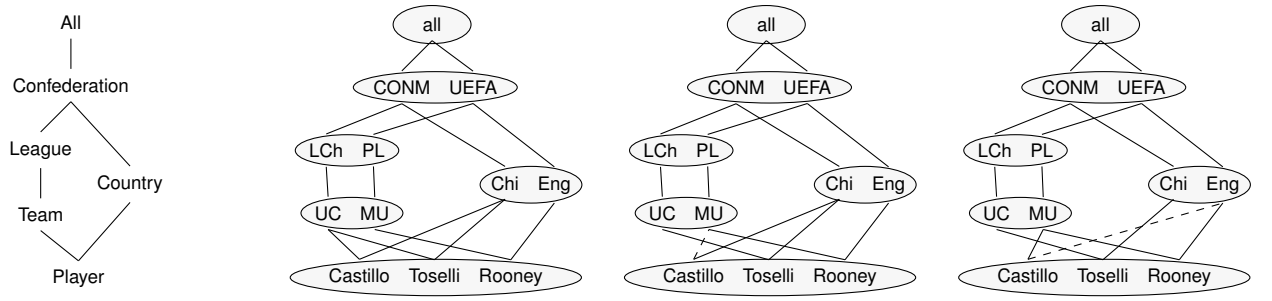
I. Introduction

Data Warehouses (DWs) integrate data from different sources, also keeping their history for analysis and decision support. DWs represent data according to *dimensions* and *facts*. The former are modeled as hierarchies of sets of elements, where each element belongs to a category from a *hierarchy schema*. The facts correspond to events which are usually associated to numeric values known as *measures*, and are referenced using the dimension elements. As an illustration, Figure 1(a) shows the hierarchy schema of a *Football Players* dimension. Figure 1(b) shows the elements and their rollup relations. Here, CONM (that stands for Conmebol) and UEFA are elements of the category Confederation, LCh (Chilean league) and PL (Premier league) are elements of League, UC (Universidad Católica) and MU (Manchester United) are elements of Team, Chi (Chile) and Eng (England) are elements of the category Country, and finally Castillo, Toselli, and Rooney are elements of Player. The hierarchical structure of dimensions allows the computation of queries at different levels of granularity, for instance, for the dimension in Figure 1(b) it is easy to compute queries grouped by Player, League, Confederation and so on. It is a common practice in DWs to use pre-computed results at lower levels of the hierarchy to compute results at higher levels. This capability is called *summarizability* [1], [2]. To guarantee summarizability, a dimension must satisfy some constraints. First, it must be *strict*, that is, every element of a category should reach (i.e., roll-up to) no more that one element in each ancestor category. Second, the dimension must be *covering*, which means that every element of a dimension category rolls-up to some element in every ancestor category. The dimension in Figure 1(b) is strict and covering. Strictness and covering *constraints* can enforce these properties in dimensions [1], [3]. It has been shown that dimensions need to be updated to adapt to changes in data sources or even to correct errors [4]. Since, current commercial DW systems do not enforce strictness, a dimension may become inconsistent with respect to its constraints after updates operations. Thus, ensuring consistency of dimensions is crucial for efficient query answering. As an illustration, suppose that in the dimension in Figure 1(b), the player Castillo is now going to play in MU instead of in UC. The DW administration must perform a $Reclassification$ of edges. After this update, the dimension violates the strictness constraint Player→Confederation, that establishes that a player must be associated with a unique confederation, since now the player Castillo goes to two different elements in Confederation, it reaches CONM via Chi in the Country category, and UEFA through element MU in the Team category (see Figure 1(c)). To fix this problem, the dimension needs to be corrected.

A *minimal r-repair* for a dimension $\mathcal{D}$ is a new dimension that is obtained from $\mathcal{D}$ by performing a minimum number of insertions and deletions of edges between elements, and keeps the reclassifications [5]. Dimension in Figure 1(d) is an r-repair for the inconsistent dimension in Figure 1(c). It was obtained by deleting the edge (Castillo,Chi) and inserting edge (Castillo,Eng). In [5] it was shown that in the general case, finding a *minimal r-repair* is NP-hard. However, under certain conditions computing r-repairs can be done in polynomial time.

II. Computing r-repairs

We present algorithms to compute r-repairs in polynomial time for the class of dimensions with at most one conflicting level [4]. Intuitively, a dimension that can lead to non-strict paths, as the football player dimension where the single conflicting category is Confederation. Dimensions become inconsistent with respect to strictness after one reclassification of edges. The algorithms follow two heuristics such that the distance between the *r-repair* obtained and the minimal *r-repair* is bound. The first heuristics ensures that when choosing a repair operation we do not generate new non-strict paths.



(a) Players dimension hierarchy (b) Elements and rollup relations (c) Inconsistent dimension $\mathcal{D}$ (d) r-repair $\mathcal{D}'$ for dimension $\mathcal{D}$

Fig. 1. Football Players Dimension

The second heuristics is aimed at guaranteeing that at each step the algorithm chooses a repair operation that requires the least number of changes.

Algorithm 1 receives as input the conflicting category ($CatCL$), the inconsistent dimension $\mathcal{D}$ and the reclassification of edges R that left the dimension inconsistent. It first gets all the inconsistent paths (from the bottom category to the conflicting category), and for all of them, the old and new parent in the conflicting category, this is, the parents before and after the reclassification. Then, for each inconsistent element, the algorithm obtains the number of paths that rollup to the new parent and the number of paths that reach the old parent in the conflicting category. Then, in order to follow the second heuristics, for every inconsistent element, the following decisions are taken: (i) if there is an equal number of paths reaching the old and the new parent in the conflicting category, the algorithm tries to keep the new parent in that category (line 1.5). (ii) If the number of paths that reach the old parent is greater than the number of paths that rollup to the new parent, the algorithm tries to restore consistency by performing reclassifications of edges in such a way that the inconsistent paths reach the old parent (line 1.8). It can be shown that the algorithm does not remove the reclassification that produces the inconsistency, and that always finds a way to restore consistency. As an illustration, for the inconsistent dimension in Figure 1(c) the inconsistent paths are Castillo-MU-PL-UEFA and Castillo-Chi-CONM, in this case, the number of paths that reach the new (UEFA) and old parent (CONM) in Confederation is equal, then the algorithm tries to restore consistency by assigning to the second path the element UEFA in category Confederation. In this case, it is not possible to change the parent of Chi to UEFA since this reclassification will produce new inconsistencies. Then, the algorithm tries to change the parent of element Castillo to Eng in the category Country. Since, this change does not produce new inconsistencies, and the edge (Castillo,Chi) is not involve in the reclassification, the algorithm produces the r-repair in Figure 1(d).

We perform experiments in a Linux machine with Debian 7.1 operative system 64 bits, with 8 GB of RAM, 640 GB of hard disk and an Intel core i5 processor of 2.4 Ghz. We use an hierarchy schema that encode the Chilean's phone system, with bottom category Number that goes to AreaCode and to City, and the latter goes to Region, which reaches All. The category Number has 2000000 elements (phone numbers), there are 346 elements in City, 27 elements in AreaCode, and 15 elements in Region. We perform certain reclassifications that change the area code to a different region and produce different levels of inconsistencies. The performance of the algorithm can be see in Figure 2 where we consider until 10% of inconsistencies which is a high level of dirty data.

Algorithm 1: Computation of r-repairs()

Input: $CatCL$, $\mathcal{D}$, R
Output: r-repair $\mathcal{D}'$

```
1.1  inconsistents_paths ← get_Inconsistent_paths(CatCL, D);
1.2  newfather ← Find_new_father(Inconsistents_paths, R);
1.3  oldfather ← Find_old_father(Inconsistents_paths, newfather);
1.4  while Inconsistents_paths ≠ NULL do
1.5      if paths_father(newfather) = paths_father(oldfather) OR flag=0 then
1.6          aux_path ← inconsistents_paths;
1.7          flag = find_repair(aux_path,new_father,old_father);
1.8      if paths_father(oldfather) > paths_father(newfather) OR flag=0 then
1.9          aux_path ← inconsistents_paths;
1.10         flag = find_repair(aux_path,old_father,new_father);
1.11     if flag ≠ 0 then
1.12         flag ← 0;
1.13         for i ← 1 to child_number do
1.14             Inconsistents_paths ← Inconsistents_paths → next;
```

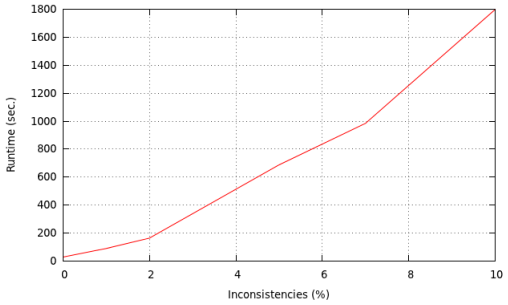


Fig. 2. Performance of the algorithm to compute an r-repair

REFERENCES

[1] C. Hurtado, C. Gutierrez, and A. Mendelzon, "Capturing Summarizability with Integrity Constraints in OLAP," *ACM Transactions on Database Systems*, vol. 30, no. 3, pp. 854–886, 2005.

[2] H.-J. Lenz and A. Shoshani, "Summarizability in OLAP and Statistical Data Bases," in *SSDBM'97*, 1997, pp. 132–143.

[3] M. Caniupán, L. Bravo, and C. A. Hurtado, "Repairing inconsistent dimensions in data warehouses," *Data Knowl. Eng.*, vol. 79-80, pp. 17–39, 2012.

[4] C. Hurtado, A. Mendelzon, and A. Vaisman, "Updating OLAP Dimensions," in *DOLAP'99*, 1999, pp. 60–66.

[5] M. Caniupán and A. Vaisman, "Repairing Dimension Hierarchies under Inconsistent Reclassification," in *MOREBI'11*, ser. Springer LNCS 6999, 2011, pp. 75–85.