Extended Dimensions for Cleaning and Querying Inconsistent Data Warehouses

Juan Ramírez
Universidad del Bío-Bío
Concepción, Chile
juaramir@ubiobio.cl

Loreto Bravo
Universidad de Concepción
Concepción, Chile
lbravo@udec.cl

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

ABSTRACT

A dimension in a data warehouse (DW) is an abstract concept that groups data that share a common semantic meaning. The dimensions are modeled using a hierarchical schema of categories. A dimension is called *strict* if every element of each category has exactly one ancestor in each parent category, and *covering* if each element of a category has an ancestor in each parent category. If a dimension is strict and covering we can use pre-computed results at lower levels to answer queries at higher levels. This capability of computing summaries is vital for efficiency purposes. Nevertheless, when dimensions are not strict/covering it is important to know their strictness and covering constraints to keep the capability of obtaining correct summarizations. Real world dimensions might fail to satisfy these constraints, and, in these cases, it is important to find ways to *fix* the dimensions (correct them) or find ways to get correct answers to queries posed on inconsistent dimensions. A *minimal repair* is a new dimension that satisfies the *strictness* and *covering* constraints, and that is obtained from the original dimension through a minimum number of changes. The set of minimal repairs can be used as a tool to compute answers to aggregate queries in the presence of inconsistencies. However, computing all of them is NP-hard. In this paper, instead of trying to find all possible minimal repairs, we define a single *compatible repair* that is consistent with respect to both strictness and covering constraints, is close to the inconsistent dimension, can be computed efficiently and can be used to compute approximate answers to aggregate queries. In order to define the compatible repair we defined the notion of extended dimension that supports sets of elements in categories.

Categories and Subject Descriptors

H.2 [**Database Management**]: Database Administration—*Data warehouse and repository*

Keywords

Data Warehouse, inconsistency, strictness constrains, covering constraints

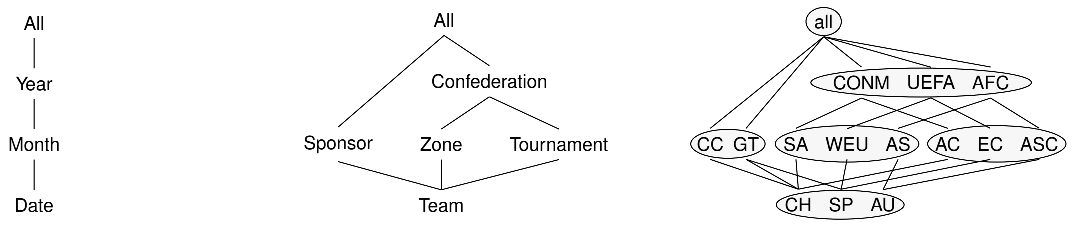
1. INTRODUCTION

Data Warehouses (DWs) are data repositories that integrate data from different sources, and keep historical data for analysis and decision support [7]. DWs organize data according to the multidimensional model. In this model, dimensions reflect the perspectives upon which facts are viewed, and the facts correspond to events which are usually associated to numeric values known as *measures*, and are referenced using the dimension elements. Dimensions are modeled as hierarchies of elements, where each element belongs to a category. The categories are also organized into a hierarchy called hierarchy schema.

Example 1. FIFA[1] has a DW to manage information about national football teams. This DW uses a Time and Football_Team dimensions. The Time dimension contains categories Date, Month, Year and All connected as shown in Figure 1(a). The Football_Team dimension, shown in Figure 1(b), contains in the *bottom* category Team the names of national football teams, their Sponsor, Zone (geographical zone) and Tournament (different important competitions such as the American Cup codified by AC). The categories Zone and Tournament are connected to Confederation category that contains associations such as Union of European Football Associations (UEFA). Figure 1(c) shows the elements for categories of the dimension Football_Team, and the *rollup relations* between them. For instance, CH (the Chilean football team), SP (the Spanish football team), AU (the Australian football team), are elements of category Team and SA (South America), WEU (Western Europe), and AS (Asia) are elements of category Zone. The rollup relation between these categories has the pairs (CH, SA), (SP, WEU) and (AU, AS).[2] The fact table Sales(Team, Date, Total) stores the incomes for the concept of sales for each team at a specific date. The multidimensional structure allows users to compute queries at different levels of granularity. For instance, it is easy to compute summaries such as: total income grouped by zone and month, or total income per confederation in a specific year, etc. □

Efficient query answering in DWs relies in the use of pre-computed results at lower categories to compute aggregates at higher levels in dimensions hierarchies. In a consistent summarization, each measure in the fact table can be aggregated at most once. In order to ensure that this property holds when using pre-computed results, the dimension must be *strict* and *covering* [30, 21, 16, 24]. A dimension is strict if all its rollup relations are many-to-one relations. The dimension is covering if all the rollup relations are such that they connect all the elements from the lower category to at least one element in every ancestor category. Ideally all the relations within

[1]FIFA stands for Fédération Internationale de Football Association.
[2]For the FIFA, Australia is considered as a country of Asia.



(a) Time Dimension Hierarchy (b) Football_Team Dimension Hierarchy (c) Football_Team elements and rollup relations

Figure 1: Football Teams Dimensions

a category should satisfy this property to ensure efficient computation of summaries. However, when modeling a certain domain, it might be the case that it is not reasonable for the relation to satisfy them.

Example 2. In the Football_Team dimension, shown in Figure 1(b) and (c), the rollup relation between categories Team and Sponsor is not strict, since a football team might have more than one sponsor, such as the element CH that has two sponsors. However, the rollup relation between categories Team and Zone is strict, since a football team is expected to be associated to a unique geographical zone. On the other hand, the rollup relation between Team and Sponsor is not required to be covering since a team might have no sponsor, such as the element AU. □

Since in real world applications dimensions are not always strict and covering, the flexibility to represent non-strict and non-covering rollup relations has been incorporated as a central feature of several dimension models [16, 25, 28, 29]. However, in order to guarantee the ability to compute consistent summarizations, it is important to know which relations are indeed strict and covering [15, 22]. These properties can be checked and enforced with strictness and covering constraint (see Section 2). A dimension is said to be *consistent* if it satisfies all its strictness and covering constraints. Otherwise, the dimension is inconsistent [5]. Checking if a dimension satisfies its constraints can be done in polynomial time [5].

A dimension may become inconsistent with respect to its constraints after update operations [5, 6] which are needed to adapt dimensions to changes in the data sources or even to correct errors in the original data [19, 18, 22, 13, 14, 31, 32, 33, 20, 10]. Inconsistent dimensions should be fixed, corrected or repaired, in order to ensure the correct computation of queries. Intuitively, a repair for a dimension $\mathcal{D}$ with respect to a set of strictness and covering constraints Σ is a new dimension that satisfies Σ and that is obtained by inserting and deleting edges to the original dimension [3, 5]. A repair is *minimal* if it is obtained by applying a minimal number of changes over the original dimension. Repairs can be given to DW's administrators as options to correct data errors and materialize the most suitable one. Moreover, repairs are used as an auxiliary concept for consistent query answering (CQA). Intuitively, a consistent answer to an aggregate query with grouping of attributes is a range for each group that contains the aggregation values obtained from all the minimal repairs [3].

In [5] it was proven that there are cases where the number of minimal repairs can be exponential in the number of elements of the dimension. Moreover, complexity results shown that the problem of computing a minimal repair is NP-hard, and deciding whether a dimension $\mathcal{D}'$ is a minimal repair of a dimension $\mathcal{D}$ is co-NP-complete [5]. Due to the complexity results, in the general case, computing minimal repairs of dimensions that are inconsistent with respect to a set of covering and strictness constraints is not tractable. However, as proposed in [5], it is possible to use Datalog programs under stable model semantics with negation and weak constraints to represent and compute the minimal repairs of a dimension. Even though this logic method computes all the possible minimal repairs for a dimension $\mathcal{D}$ with respect to a set of constraints Σ, its application over large sets of data is not feasible [5].

In this paper we define a single repair called the *compatible repair* that is obtained from the inconsistent dimension by performing insertions of elements and edges and deletions of edges. This repair differs from the *canonical repair* presented in [3] which requires the computation of all the minimal repairs of a dimension. In the case of compatible repairs, we want to represent in a single dimension the fact that a certain elements' rollup relation might need to be modified to restore consistency and therefore it is not known to which element in an upper category it rolls-up. In order to represent this uncertainty in a compatible repair, we introduce the concept of *extended* dimension, that is a new dimension where categories contain *sets* of elements. In this way a rollup relation between sets s_1 and s_0 implies that all rollup combinations between elements in s_0 and s_1 can possibly be valid. This ambiguity allows to represent the unknown ways in which the inconsistent dimension can be repaired and allows to provide ranges of answers between the aggregate values that are known to be part of the answer and values that might be part of it. Thus, the answers from an extended dimension will not be exact but a range within which the answer is known to belong.

The rest of the paper is organized as follows: Section 2 presents dimensions, constraints and repairs. Section 3 presents the extended dimension and illustrates how to perform query processing on these dimensions. Section 4 presents the compatible repair. Section 5 concludes this article.

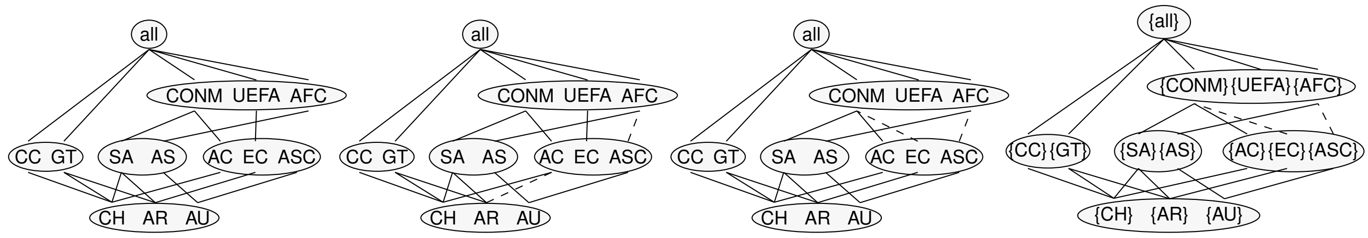
2. DIMENSIONS AND REPAIRS

The definitions here are slightly modified from the ones in [5] to simplify the presentation of the results in this paper.

A *hierarchy schema* $\mathcal{H}$ is a pair $(\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$, where $(\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$ is an acyclic directed graph. Vertices in the set $\mathcal{C}_\mathcal{H}$ are categories and the edges $\nearrow_\mathcal{H}$ represent the child/parent relations between categories. The transitive and reflexive closure of $\nearrow_\mathcal{H}$ is denoted by $\nearrow^*_\mathcal{H}$. The set of categories $\mathcal{C}_\mathcal{H}$ contains a distinguished *top* category denoted $\mathsf{All}_\mathcal{H}$, which is reachable from every other category in $\mathcal{C}_\mathcal{H}$ and has no outgoing edges. Sometimes, we will use $c_a \nearrow_\mathcal{H} c_b$ instead of $(c_a, c_b) \in \nearrow_\mathcal{H}$. To simplify presentation we assume that categories do not have attributes and that they have only one bottom category, this is a category with only outgoing edges.

A *dimension* $\mathcal{D}$ is a tuple $(\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Elem}_\mathcal{D}, <_\mathcal{D})$, where $\mathcal{H}_\mathcal{D} = (\mathcal{C}_{\mathcal{H}_\mathcal{D}}, \nearrow_{\mathcal{H}_\mathcal{D}})$ is a hierarchy schema; $\mathcal{E}_\mathcal{D}$ is a set of constants, called elements; $\mathsf{Elem}_\mathcal{D} : \mathcal{C}_{\mathcal{H}_\mathcal{D}} \to \mathcal{P}(\mathcal{E}_\mathcal{D})$[3] is a function that re-

[3]$\mathcal{P}(S)$ is the powerset of set S.



(a) Inconsistent dimension $\mathcal{D}_{\text{FT}}$ (b) Minimal repair $\mathcal{D}^1_{\text{FT}}$ (c) Minimal repair $\mathcal{D}^2_{\text{FT}}$ (d) Compatible repair $\mathcal{X}_{\text{FT}}$

Figure 2: Inconsistent dimension and its minimal repairs

turns the elements that belong to a category; and the relation $<_{\mathcal{D}} \subseteq \mathcal{E}_{\mathcal{D}} \times \mathcal{E}_{\mathcal{D}}$ represents the child/parent relation between elements of different categories. We denote by $<^*_{\mathcal{D}}$ the reflexive and transitive closure of $<_{\mathcal{D}}$. The following conditions hold: (i) $\mathsf{all}_{\mathcal{D}}$ is the only element in category $\mathsf{All}_{\mathcal{H}_{\mathcal{D}}}$. (ii) For all $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}_{\mathcal{D}}}$, if $\mathsf{c}_i \neq \mathsf{c}_j$ then $\mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_i) \cap \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_j) = \emptyset$. (iii) For all pair of elements $a, b \in \mathcal{E}_{\mathcal{D}}$ if $a <_{\mathcal{D}} b$ then there exists $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}_{\mathcal{D}}}$ such that $\mathsf{c}_i \nearrow_{\mathcal{H}} \mathsf{c}_j$, $a \in \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_i)$ and $b \in \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_j)$. Condition (iii) ensures that the child/parent relation ($<_{\mathcal{D}}$) only connects elements of categories that are connected in the schema. In this definition of dimension we use the function $\mathsf{Elem}_{\mathcal{D}}$ to obtain the elements of categories instead of having a function that returns the category of an specific element as in [15, 5].

Example 3. The hierarchy schema $\mathcal{H}_{\text{FT}} = (\mathcal{C}_{\text{FT}}, \nearrow_{\text{FT}})$ in Figure 1(b) is formalized as follows: $\mathcal{C}_{\text{FT}} = \{$Team, Sponsor, Zone, Tournament, Confederation, All$\}$, and $\nearrow_{\text{FT}} = \{$(Team, Sponsor), (Team, Zone), (Team, Tournament), (Sponsor, All),(Zone, Confederation), (Tournament, Confederation), (Confederation, All)$\}$.

The dimension $\mathcal{D}_c = (\mathcal{H}_{\text{FT}}, \mathcal{E}_c, \mathsf{Elem}_c, <_c)$ given in Figure 1(c) is formalized as follows:
(i) $\mathcal{E}_c = \{$ all, CONM, UEFA, AFC, CC,GT, SA, WEU, AS, AC, EC, ASC, CH, SP, AU$\}$;
(ii) Elem$_c$(Team)= $\{$CH, SP, AU$\}$, Elem$_c$(Sponsor)= $\{$CC, GT$\}$, Elem$_c$(Zone)= $\{$SA, WEU, AS$\}$, Elem$_c$(Tournament)= $\{$AC, EC, ASC$\}$, Elem$_c$(Confederation)= $\{$CONM, UEFA, AFC$\}$, Elem$_c$(All)= $\{$all$\}$; (iii) $<_c = \{$(CH, CC), (CH, GT), (CH, SA), (CH, AC), (SP, GT), (SP, WEU), (SP, EC), (AU, AS), (AU, ASC), (CC, all), (GT, all), (SA, CONM), (WEU, UEFA), (AS, AFC), (AC, CONM), (EC, UEFA), (ASC, AFC), (CONM, all), (UEFA, all), (AFC, all)$\}$. □

For each pair of categories $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}_{\mathcal{D}}}$ such that $\mathsf{c}_i \nearrow^*_{\mathcal{H}_{\mathcal{D}}} \mathsf{c}_j$, there is a rollup relation denoted by $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ that has the following set of pairs $\{(a, b) | a \in \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_i), b \in \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_j)$ and $a <^*_{\mathcal{D}} b\}$. For example, the rollup relation between categories Team and Zone denoted with $\mathcal{R}_c$(Team, Zone) has the pairs: $\{$(CH,SA), (SP, WEU), (AU, AS)$\}$.

Let $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ be a rollup relation, then: (i) $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ is *strict* if $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ is a function, i.e., if for all elements x, y, z in $\mathcal{E}_{\mathcal{D}}$, if $(x, y) \in \mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ and $(x, z) \in \mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ then $y = z$. (ii) $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ is *covering* if for all elements $e \in \mathcal{E}_{\mathcal{D}}$ such that $e \in \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_i)$, there exists an element $e' \in \mathcal{E}_{\mathcal{D}}$ such that $e' = \mathsf{Elem}_{\mathcal{D}}(\mathsf{c}_j)$ and $(e, e') \in \mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$.

In the majority of the research and industrial OLAP applications dimensions are considered to satisfy all possible strict and covering constraints that can be defined for the hierarchy of the dimension [9]. Nevertheless, dimensions might fail to satisfy these conditions [26, 28, 17, 16, 23, 15]. When this happens it is necessary to specify integrity constraints to identify rollup relations that are strict or covering, and in this way to be able of keeping the capability of computing correct summarizations [17].

Let $\mathcal{H} = (\mathcal{C}_{\mathcal{H}}, \nearrow_{\mathcal{H}})$ be a hierarchy schema and let $\mathcal{D} = (\mathcal{H}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}}, \mathsf{Elem}_{\mathcal{D}}, <_{\mathcal{D}})$ be a dimension such that $\mathcal{H}_{\mathcal{D}} = \mathcal{H}$. (i) A *strictness constraint* over $\mathcal{H}$ is an expression of the form $\mathsf{c}_i \rightarrow \mathsf{c}_j$ where $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}}$ and $\mathsf{c}_i \nearrow^*_{\mathcal{H}} \mathsf{c}_j$. The dimension $\mathcal{D}$ *satisfies* the strictness constraint $\mathsf{c}_i \rightarrow \mathsf{c}_j$ if and only if the rollup relation $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ is strict. (ii) A *covering constraint* over $\mathcal{H}$ is an expression of the form $\mathsf{c}_i \Rightarrow \mathsf{c}_j$ where $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}}$ and $\mathsf{c}_i \nearrow^*_{\mathcal{H}} \mathsf{c}_j$. The dimension $\mathcal{D}$ *satisfies* the covering constraint $\mathsf{c}_i \Rightarrow \mathsf{c}_j$ if and only if the rollup relation $\mathcal{R}_{\mathcal{D}}(\mathsf{c}_i, \mathsf{c}_j)$ is covering. A dimension $\mathcal{D}$ satisfies Σ if $\mathcal{D}$ satisfies every constraint in Σ. Otherwise, the dimension $\mathcal{D}$ is inconsistent with respect to Σ [5].

Example 4. (example 1 continued) The Football_Team dimension $\mathcal{D}_c$=($\mathcal{H}_{\text{FT}}$, $\mathcal{E}_c$, Elem$_c$, $<_c$) is consistent with respect to the following set of strictness and covering constraints: $\Sigma = \{$Team $\rightarrow$ Zone, Team $\rightarrow$ Tournament, Zone $\rightarrow$ Confederation, Tournament $\rightarrow$ Confederation, Team $\rightarrow$ Confederation, Team $\Rightarrow$ Zone, Team $\Rightarrow$ Tournament, Zone $\Rightarrow$ Confederation, Tournament $\Rightarrow$ Confederation$\}$. On the other hand, the dimension $\mathcal{D}_{\text{FT}}$ in Figure 2(a) is not consistent with respect to Σ. In particular $\mathcal{D}_{\text{FT}}$ violates the strictness constraint Team $\rightarrow$ Confederation since the element AR goes to elements CONM and UEFA in category Confederation. Also, $\mathcal{D}_{\text{FT}}$ does not satisfy the covering constraint Tournament $\Rightarrow$ Confederation since the element ASC in the Tournament category is not connected with an element in the Confederation category. □

When a dimension does not satisfy its constraints it needs to be corrected (repaired). In order to define repairs we need the concept of distance: Given two dimensions $\mathcal{D} = (\mathcal{H}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}}, \mathsf{Elem}_{\mathcal{D}}, <_{\mathcal{D}})$ and $\mathcal{D}' = (\mathcal{H}_{\mathcal{D}'}, \mathcal{E}_{\mathcal{D}'}, \mathsf{Elem}_{\mathcal{D}'}, <_{\mathcal{D}'})$, the distance between them, $dist(\mathcal{D}, \mathcal{D}')$, is defined as $|(<_{\mathcal{D}'} \setminus <_{\mathcal{D}}) \cup (<_{\mathcal{D}} \setminus <_{\mathcal{D}'})|$. In other words, the distance between two dimensions is the size of the symmetric difference between the child/parent relations of the two dimensions.

Let $\mathcal{D} = (\mathcal{H}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}}, \mathsf{Elem}_{\mathcal{D}}, <_{\mathcal{D}})$ be a dimension and Σ be a set of integrity constraints over $\mathcal{H}_{\mathcal{D}}$. (i) A *repair* of $\mathcal{D}$ with respect to Σ is a dimension $\mathcal{D}' = (\mathcal{H}_{\mathcal{D}'}, \mathcal{E}_{\mathcal{D}'}, \mathsf{Elem}_{\mathcal{D}'}, <_{\mathcal{D}'})$ such that $\mathcal{H}_{\mathcal{D}'} = \mathcal{H}_{\mathcal{D}}$, $\mathcal{E}_{\mathcal{D}'} = \mathcal{E}_{\mathcal{D}}$, $\mathsf{Elem}_{\mathcal{D}'} = \mathsf{Elem}_{\mathcal{D}}$, and $\mathcal{D}'$ satisfies Σ. (ii) A *minimal* repair of $\mathcal{D}$ with respect to Σ is a repair $\mathcal{D}'$, such that $dist(\mathcal{D}, \mathcal{D}')$ is minimal among all the repairs of $\mathcal{D}$ with respect to Σ.

Example 5. Figure 2(b)-(c) shows the two minimal repairs for the inconsistent dimension $\mathcal{D}_{\text{FT}}$ in Figure 2(a). Repair $\mathcal{D}^1_{\text{FT}}$ is obtained by inserting the edge (AR, AC), and deleting the edge (AR, EC), this change restores consistency of the strictness constraint Team$\rightarrow$ Confederation. The edge (ASC, AFC) is inserted in $\mathcal{D}^1_{\text{FT}}$ in order to satisfy the covering constraint Tournament $\Rightarrow$ Confederation. Repair $\mathcal{D}^2_{\text{FT}}$ is obtained by inserting the edges (EC, CONM) and (ASC, AFC), and deleting the edge (EC, UEFA). Both repairs are obtained by applying three changes. There are other possible repairs but they are not minimal. $\square$

Note that in the repairs, the set of elements in each category of the original dimension is preserved over all the repairs. In [5] it was proven that: (i) there always exists a repair for a given dimension $\mathcal{D}$ (it is necessary that each category has at least one element), (ii) the number of minimal repairs can be exponential in the number of elements of the dimension, (iii) computing a minimal repair for a dimension $\mathcal{D}$ is NP-hard.

Minimal repairs can be used to compute consistent answers to queries [1, 3]. To obtain a consistent answer we need to compute every answer in every possible minimal repair. Thus, given these complexity results, it is of importance to find ways to compute approximate answers that take into consideration the existence of inconsistencies in the data. In the next section we present the extended dimension that allow us to introduce sets of elements in categories.

3. EXTENDED DIMENSION

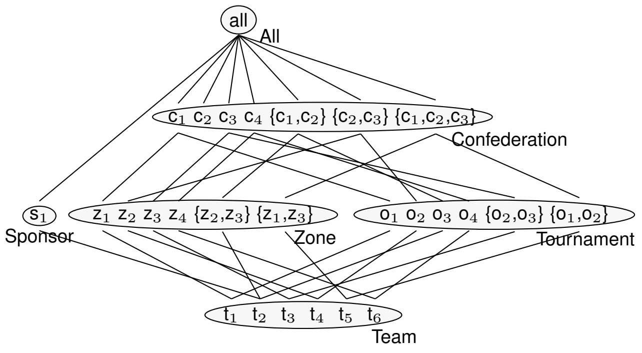
When a dimension is inconsistent with respect to its constraints some how there are elements that present ambiguity with respect to their rollups. For instance a violation of a strictness constraint $c_i \rightarrow c_j$ is produced when there is an element in category c_i reaches more than one element in category c_j, thus there is error or imprecision in the data. Uncertainty has been analysed in the context of multidimensional models in [27, 4].

We will repair data inconsistencies by inserting new elements in categories, thus, we need to formalize the class of dimensions that support sets of elements in categories.

Definition 1. An *extended dimension* $\mathcal{X}$ is a tuple $(\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$, where $\mathcal{H}_\mathcal{X} = (\mathcal{C}_{\mathcal{H}_\mathcal{X}}, \nearrow_{\mathcal{H}_\mathcal{X}})$ is a hierarchy schema; $\mathcal{E}_\mathcal{X}$ is a set of constants, called elements; $\mathsf{CElem}_\mathcal{X} : \mathcal{C}_{\mathcal{H}_\mathcal{X}} \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{E}_\mathcal{X}))$ is a function that, given a category returns a set of subsets of elements in $\mathcal{E}_\mathcal{X}$; and the relation $\ll_\mathcal{X} \subseteq \mathcal{P}(\mathcal{E}_\mathcal{X}) \times \mathcal{P}(\mathcal{E}_\mathcal{X})$ that corresponds to the child/parent relation between elements of different categories. We denote with $\ll^*_\mathcal{X}$ the reflexive and transitive closure of $\ll_\mathcal{X}$. The following conditions hold: (i) $\mathsf{all}_\mathcal{X}$ is the only element in category $\mathsf{All}_{\mathcal{H}_\mathcal{X}}$. (ii) For all $c_i, c_j \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$, if $c_i \neq c_j$ then $\mathsf{CElem}_\mathcal{X}(c_i) \cap \mathsf{CElem}_\mathcal{X}(c_j) = \emptyset$. (iii) For all pair of elements $a \in \mathsf{CElem}_\mathcal{X}(c_i)$ and $b \in \mathsf{CElem}_\mathcal{X}(c_j)$ if $a \ll_\mathcal{X} b$ then $c_i \nearrow_{\mathcal{H}_\mathcal{X}} c_j$. (iv) For each $c_i \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$ it holds that: (a) $\emptyset \notin \mathsf{CElem}_\mathcal{X}(c_i)$. (b) If $e \in \mathsf{CElem}_\mathcal{X}(c_i)$ then for each element $e' \in e$ it holds that $\{e'\} \in \mathsf{CElem}_\mathcal{X}(c_i)$. (v) for the bottom category $c_b \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$ it holds that for every $e \in \mathsf{CElem}_\mathcal{X}(c_b)$, e is a singleton. $\square$

Condition (iii) ensures that the child/parent relation ($\ll_\mathcal{X}$) only connects elements of categories that are connected in the schema. Condition (iv) ensures that every element that is a set in a category contains only elements of this category. Condition (v) enforces that elements in the bottom category are only singleton elements, and in this way, we will be able to join with the data in the fact tables. In order to simplify presentation we will sometimes replace a singleton element $\{e\}$ by e.

Let $\mathcal{H}_\mathcal{X} = (\mathcal{C}_{\mathcal{H}_\mathcal{X}}, \nearrow_{\mathcal{H}_\mathcal{X}})$ be a hierarchy schema and let $(\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$ be an extended dimension for the schema. For each pair of categories $c_i, c_j \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$ such that $c_i \nearrow^*_{\mathcal{H}_\mathcal{X}} c_j$, there is a rollup relation denoted by $\mathcal{R}_\mathcal{X}(c_i, c_j)$ that has the following set of pairs $\{(a, b) | a \in \mathsf{CElem}_\mathcal{X}(c_i), b \in \mathsf{CElem}_\mathcal{X}(c_j) \text{ and } a \ll^*_\mathcal{X} b\}$.



all
All
c1 c2 c3 c4 {c1,c2} {c2,c3} {c1,c2,c3}
Confederation
s1
Sponsor
z1 z2 z3 z4 {z2,z3} {z1,z3}
Zone
o1 o2 o3 o4 {o2,o3} {o1,o2}
Tournament
t1 t2 t3 t4 t5 t6
Team

Figure 3: Extended Dimension $\mathcal{X}_{\text{FT}}$

A *strictness constraint* over $\mathcal{H}_\mathcal{X}$ is an expression of the form $c_i \rightarrow c_j$ where $c_i, c_j \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$ and $c_i \nearrow^*_{\mathcal{H}_\mathcal{X}} c_j$. The extended dimension $\mathcal{D}$ *satisfies* the strictness constraint $c_i \rightarrow c_j$ if and only if for each element $e_i \in \mathsf{CElem}_\mathcal{X}(c_i)$ there are no elements $e_j, e_k \in \mathsf{CElem}_\mathcal{X}(c_j)$ such that $e_j \neq e_k$ and $e_i \ll^* e_j$ and $e_i \ll^* e_k$. A *covering constraint* over $\mathcal{H}_\mathcal{X}$ is an expression of the form $c_i \Rightarrow c_j$ where $c_i, c_j \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}$ and $c_i \nearrow^*_{\mathcal{H}_\mathcal{X}} c_j$. The dimension $\mathcal{D}$ *satisfies* the covering constraint $c_i \Rightarrow c_j$ if and only if the for every $e_i \in \mathsf{CElem}_\mathcal{X}(c_i)$ there exists an element $e_j \in \mathsf{CElem}_\mathcal{X}(c_j)$ such that $e_i \ll^* e_j$.

We say that an extended dimension $\mathcal{X}$ satisfies Σ if $\mathcal{X}$ satisfies every constraint in Σ. Otherwise, the extended dimension $\mathcal{X}$ is inconsistent with respect to Σ.

Example 6. A possible extended dimension $\mathcal{X}_{\text{FT}}$ for the hierarchy schema in Figure 1(b) is shown in Figure 3. The rollup relation between categories Team and Zone, $\mathcal{R}_\mathcal{X}(\text{Team}, \text{Zone})$, has the pairs: $\{(t_1,z_1), (t_2,\{z_2, z_3\}),(t_3, z_3), (t_4,z_2), (t_5,\{z_1,z_3\}), (t_6,z_4)\}$. Note that some of the elements in Zone are sets. The extended dimension $\mathcal{X}$ satisfies the set of constraints $\Sigma = \{$Team $\rightarrow$ Zone, Team $\rightarrow$ Tournament, Zone $\rightarrow$ Confederation, Tournament $\rightarrow$ Confederation, Team $\rightarrow$ Confederation, Team $\Rightarrow$ Zone, Team $\Rightarrow$ Tournament, Zone $\Rightarrow$ Confederation, Tournament $\Rightarrow$ Confederation$\}$.

Note that an extended dimension can be treated as a traditional dimension where all sets belonging to a category are considered as simple elements. For example, the extended dimension $\mathcal{X}_{\text{FT}}$ in Figure 3, can be seen as a traditional dimension where categories Team, Zone and Tournament contain six distinct elements and category Confederation contains seven distinct elements. However, note that this results in loosing the information about the connection between elements. For example, it would interpret that there is no connection between elements o_1 and $\{o_1, o_2\}$.

Definition 2. For an extended dimension $\mathcal{X} = (\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$, with $\mathcal{H}_\mathcal{X} = (\mathcal{C}_{\mathcal{H}_\mathcal{X}}, \nearrow_{\mathcal{H}_\mathcal{X}})$, its associated dimension, denoted $\mathcal{T}(\mathcal{X})$, is a dimension $(\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Elem}_\mathcal{D}, <_\mathcal{D})$, where:

- $\mathcal{H}_\mathcal{D} = \mathcal{H}_\mathcal{X}$,
- $\mathcal{E}_\mathcal{D} = \bigcup_{c_i \in \mathcal{C}_{\mathcal{H}_\mathcal{X}}} \mathsf{CElem}_\mathcal{X}(c_i)$,
- $\mathsf{Elem}_\mathcal{D} = \mathsf{CElem}_\mathcal{X}$ with domain in $\mathcal{E}_\mathcal{D}$, and
- $<_\mathcal{D} = \ll_\mathcal{X}$ which relates elements in $\mathcal{E}_\mathcal{D}$. $\square$

Since for an extended dimension $\mathcal{X}$ we have a dimension $\mathcal{T}(\mathcal{X})$ we could think that when posing a query we could use techniques currently implemented in DWs systems. However, as we will show, query answering needs to be redefined for extended dimensions to take full advantage of the flexibility given by extended dimensions.

	Confederation	Year	SUM	COUNT
t_1	c_1	{2010,2011}	400	1
t_2	c_1	2011	400	1
t_3	c_1	2012	200	1
t_4	c_1	{2012,2013}	300	1
t_5	$\{c_1, c_2\}$	2010	600	1
t_6	$\{c_1, c_2\}$	2011	100	1
t_7	$\{c_1, c_2\}$	{2012,2013}	200	1
t_8	$\{c_1, c_2\}$	2013	100	1
t_9	$\{c_1, c_2, c_3\}$	2010	400	1
t_{10}	$\{c_1, c_2, c_3\}$	{2010,2011}	300	1
t_{11}	$\{c_1, c_2, c_3\}$	{2012,2013}	500	1
t_{12}	$\{c_1, c_2, c_3\}$	2013	600	1
t_{13}	$\{c_2, c_3\}$	2010	100	1
t_{14}	$\{c_2, c_3\}$	2011	200	1
t_{15}	$\{c_2, c_3\}$	{2012,2013}	300	1
t_{16}	$\{c_2, c_3\}$	2013	500	1
t_{17}	c_3	{2010,2012}	300	1
t_{18}	c_3	2011	300	1
t_{19}	c_3	2012	500	1
t_{20}	c_3	2013	200	1
t_{21}	c_4	2013	300	1

Table 1: Answers to queries over dimensions $\mathcal{T}(\mathcal{X}_{\text{FT}})$ and $\mathcal{T}(\mathcal{X}_{\text{TIME}})$ and a fact table Sales

In DWs the most common aggregate queries are those that perform grouping by the values of a set of attributes, and return a single aggregate value per group. An aggregate query is of the form:

```
SELECT A_j,... A_n, f(A)
FROM T, R_i,... R_m
WHERE conditions
GROUP BY A_j,... A_n
```

A_j,... A_n are attributes of the fact table T or the rollup functions R_i, ... R_m (treated as tables), and f is a function, such as: min(A), max(A), count(A), sum(A), applied to attribute A, with $\mathsf{A} \cap \{\mathsf{A}_j, \dots \mathsf{A}_n\} = \emptyset$.

Definition 3. An answer to an aggregate query $\mathcal{Q}$ over dimensions $\mathcal{D}_1, \dots, \mathcal{D}_i, \dots, \mathcal{D}_n$ and a fact table F is a tuple of the form $\langle t_1, \dots, t_n, c\rangle$, where each t_i is an element of $\mathcal{D}_i$, and c is the value returned for the aggregate function for the group $\langle t_1, \dots, t_n\rangle$ in the fact table F. The set of answers of $\mathcal{Q}$ over dimensions $\mathcal{D}_1, \dots, \mathcal{D}_i, \dots, \mathcal{D}_n$ and a fact table F is denoted by $\mathcal{Q}(\{\mathcal{D}_1, \dots, \mathcal{D}_i, \dots, \mathcal{D}_n\}, F)$. □

Given a cube with extended dimensions $\mathcal{X}_1, \dots, \mathcal{X}_i, \dots, \mathcal{X}_n$, fact table F and query Q we could try to provide answers by simply evaluating the query over dimensions $\mathcal{T}(\mathcal{X}_i)$ for $i \in [1, n]$, namely $Q(\{\mathcal{T}(\mathcal{X}_1), \dots, \mathcal{T}(\mathcal{X}_i), \dots, \mathcal{T}(\mathcal{X}_n)\}, F)$. In the following example we show that the results obtained with this method do not allow the user to see the effect of the relationship between elements within a category.

Example 7. Consider Table 1 that shows a set of answers that can be obtained from a Sales fact table (which is not shown here because of space restrictions) when posing aggregate queries Q_j that obtain, respectively, the SUM and COUNT of incomes grouped by Confederation, and Year over dimensions $\mathcal{T}(\mathcal{X}_{\text{FT}})$, and $\mathcal{T}(\mathcal{X}_{\text{TIME}})$. It is easy to see that the results in $Q_j(\{\mathcal{T}(\mathcal{X}_{\text{FT}}), \mathcal{T}(\mathcal{X}_{\text{TIME}})\}, \mathsf{Sales})$ do not capture the fact that different elements might contribute to the same aggregation. For instance, tuple t_6 in Table 1 might contribute to the answer in tuple t_2 since c_1 is contained in element $\{\mathsf{c}_1, \mathsf{c}_2\}$. □

Since $Q(\{\mathcal{T}(\mathcal{X}_1), \dots, \mathcal{T}(\mathcal{X}_i), \dots, \mathcal{T}(\mathcal{X}_n)\}, F)$ does not capture the desired semantics of query answering from extended dimensions, we need to provide a specific formalization for extended dimensions that takes into consideration the relationship between elements within a category.

Definition 4. Given a query $\mathcal{Q}$, an extended dimension $\mathcal{X} = (\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$, a fact table F, and a tuple $t = \langle e_1, \dots, e_n\rangle$ with $e_i \in \mathcal{E}_\mathcal{X}$ for every $i \in [1, n]$, the set of *tuples associated to t in* $Q(\{\mathcal{T}(\mathcal{X}_1), \dots, \mathcal{T}(\mathcal{X}_i), \dots, \mathcal{T}(\mathcal{X}_n)\}, F)$, denoted by $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_1, \dots, \mathcal{X}_i, \dots, \mathcal{X}_n\}, F)$, is $\{a | a = \langle s_1, \dots, s_n, v\rangle, a \in Q(\mathcal{T}(\mathcal{X}_1), \dots, \mathcal{T}(\mathcal{X}_i), \dots, \mathcal{T}(\mathcal{X}_n), F), e_i \in s_i \text{ for } i \in [1, n]\}$. □

Intuitively, set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_1, \dots, \mathcal{X}_i, \dots, \mathcal{X}_n\}, F)$ contains all the tuples that might contribute to answer tuple t.

Definition 5. Given a query $\mathcal{Q}$ with aggregate function SUM or $COUNT$, a set of extended dimension $\mathcal{S} = \{\mathcal{X}_1, \dots, \mathcal{X}_i, \dots, \mathcal{X}_n\}$ where $\mathcal{X}_i = (\mathcal{H}_{\mathcal{X}_i}, \mathcal{E}_{\mathcal{X}_i}, \mathsf{CElem}_{\mathcal{X}_i}, \ll_{\mathcal{X}_i})$, and a fact table F, the *answers of $\mathcal{Q}$ over $\mathcal{X}$ and F* is denoted by $\widetilde{Q}(\mathcal{S}, F)$. A tuple $\langle e_1, \dots, e_n, [a, b]\rangle$ with $e_i \in \mathcal{E}_\mathcal{X}$ for every $i \in [1, n]$ and $[a, b]$ a numerical range belongs to $\widetilde{Q}(\mathcal{S}, F)$ if and only if: (i) $\mathcal{A}(\langle e_1, \dots, e_n\rangle, \mathcal{Q}, \mathcal{S}, F) \neq \emptyset$; (ii) one of the following holds: either $a = v$ when there exists v such that $\langle \{e_1\}, \dots, \{e_n\}, v\rangle \in Q(\{\mathcal{T}(\mathcal{X}_1), \dots, \mathcal{T}(\mathcal{X}_i), \dots, \mathcal{T}(\mathcal{X}_n)\}, F)$ or $a = 0$ when there does not exists such v; and $b = \sum_{\langle s_1, \dots, s_n, v\rangle \in V} v$ where $V = \mathcal{A}(\langle e_1, \dots, e_n\rangle, \mathcal{Q}, \mathcal{S}, F)$. □

If we consider, instead of SUM and COUNT different aggregate functions, then the definition of approximate answer should be modified since the semantics is tightly connected to it. For instance, in the case of MIN and MAX, instead of a range, we would be interested in the set of possible minimal/maximal values that correspond to the possible answers. Due to space limitations we left this formalization for an extended version.

Example 8. Table 2 shows the answers to the aggregates queries with aggregate functions SUM and COUNT over the extended dimensions $\mathcal{X}_{\text{FT}}$ and $\mathcal{X}_{\text{TIME}}$ in Figure 3, which are obtained considering the answers given in Table 1.

Confederation	Year	SUM	COUNT
c_1	2010	$[0, 1700]$	$[0, 4]$
c_1	2011	$[400, 1200]$	$[1, 4]$
c_1	2012	$[200, 1200]$	$[1, 4]$
c_1	2013	$[0, 1700]$	$[0, 5]$
c_2	2010	$[0, 1400]$	$[0, 4]$
c_2	2011	$[0, 600]$	$[0, 3]$
c_2	2012	$[0, 1000]$	$[0, 3]$
c_2	2013	$[0, 2200]$	$[0, 6]$
c_3	2010	$[0, 1100]$	$[0, 4]$
c_3	2011	$[300, 800]$	$[1, 3]$
c_3	2012	$[500, 1600]$	$[1, 4]$
c_3	2013	$[200, 2100]$	$[1, 5]$
c_4	2013	$[300, 300]$	$[1, 1]$

Table 2: Answers to an aggregate query over extended dimensions $\mathcal{X}_{\text{FT}}$ and $\mathcal{X}_{\text{TIME}}$

As an illustration, for tuple $t = \langle c_1, 2010\rangle$ set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$ contains tuples t_1, t_5, t_9 and t_{10} (see Table 1), in this case, there is no answer for t in Table 1, thus the final answer for the aggregate query with aggregate function SUM is $\langle c_1, 2010, [0, 1700]\rangle$, where 1700 is the sum of the aggregate values for tuples in the set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$. For tuple $t = \langle c_1, 2011\rangle$ set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$ contains the tuples t_1, t_2, t_6 and t_{10}, since there is an answer for t in Table 1 (tuple t_2), the final answer for SUM is $\langle c_1, 2011, [400, 1200]\rangle$ where value 400 is the answer for t given by tuple t_2 and 1200 is the sum of the aggregate values for tuples in $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$. For the tuple $t = \langle c_4, 2010\rangle$ set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$ is empty, since there is no answer for t in Table 1, then, there is no answer for this group (and neither for c_4 with years 2011 and 2012). However, for $t = \langle c_4, 2013\rangle$ the set $\mathcal{A}(t, \mathcal{Q}, \{\mathcal{X}_{\text{FT}}, \mathcal{X}_{\text{TIME}}\}, \mathsf{Sales})$ has the unique tuple t_{21} in Table 1, then, the final answer for SUM is $\langle c_4, 2013, [300, 300]\rangle$ where value 300 is the answer for $t = \langle c_4, 2013\rangle$ in Table 1. An analogous analysis can be done for the aggregate function COUNT. □

4. COMPATIBLE REPAIR

In this section we present the concept of *compatible repair* that is a new extended dimension obtained from the inconsistent dimension that is consistent with respect to the set of constraints.

The idea behind the compatible repair is to be able of having a unique repair, to be used to compute approximate answers to aggregate queries. The definition of the compatible repair is inspired in the method presented in [26] to obtain strict dimensions. According to this method, strictness is restored by inserting new fused elements into artificial categories. As an example, if an element a rolls-up to both b and c in a category D, the element {b,c} is created in a new category and a is associated with this new composed element. Also, links are created from the fused element to the original parents of a in the conflicting category D. We only create new elements in conflicting categories when there is no other option to restore strictness. To formalize the concept of compatible dimension we use the *extended dimension* presented in Section 3, which is a dimension that might have set of elements in categories.

We also need the concept of *conflicting category* which was introduced in [19]: Given an extended dimension $(\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$ over a schema $\mathcal{H}_\mathcal{X} = (\mathcal{C}_{\mathcal{H}_\mathcal{X}}, \nearrow_{\mathcal{H}_\mathcal{X}})$, a pair of categories c_i and c_j such that $\mathsf{c}_i \nearrow_{\mathcal{H}_\mathcal{X}} \mathsf{c}_j$, a pair of elements a, b with $a \in \mathsf{CElem}_\mathcal{X}(\mathsf{c}_i)$ and $b \in \mathsf{CElem}_\mathcal{X}(\mathsf{c}_j)$. A category c_k such that $\mathsf{c}_j \nearrow^*_{\mathcal{H}_\mathcal{X}} \mathsf{c}_k$ is a *conflicting* category, if there exists a category c_m such that $\mathsf{c}_m \nearrow^*_{\mathcal{H}_\mathcal{X}} \mathsf{c}_i$, there is an alternative path between c_m and c_k not including the edge $(\mathsf{c}_i,\mathsf{c}_j)$, and a is reached by at least one element in c_m. As an illustration, the category Confederation in the hierarchy schema in Figure 1(a) is conflicting since there are alternative paths from category Team that reach Confederation. One path starts in Team going to Zone and finally reaching Confederation, the other path starts in Team going to Tournament and then to Confederation. In this case, we have c_i =Zone, $\mathsf{c}_j = \mathsf{c}_k$ =Confederation, and c_m =Team. Edges (Team,Tournament) and (Tournament,Confederation) form an alternative path from Team to Confederation not including the edge (Zone,Confederation).

Definition 6. Given a dimension $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Elem}_\mathcal{D}, <_\mathcal{D})$ which is inconsistent with respect to a set Σ of strictness and covering constraints. Dimension $\mathcal{X} = (\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$ is the compatible repair of $\mathcal{D}$ with respect to Σ if it is obtained through the Algorithm 1. □

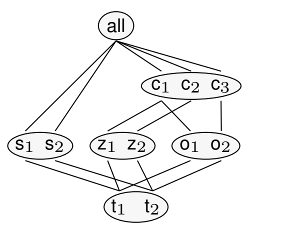
The compatible repair is obtained from the inconsistent dimension by inserting new elements and edges and deleting edges. To perform changes over the dimension it is important to analyze the existence of *evidence* of consistency in the inconsistent dimension. Algorithm 1 receives the set Σ of strictness and covering constraints,

Algorithm 1: Algorithm to Compute the Compatible Repair

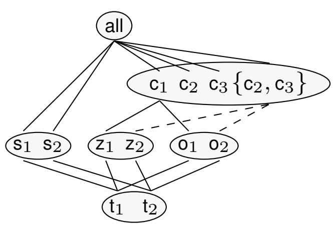
```
     Input: $\Sigma, \mathcal{D}$
     Output: Compatible Repair $\mathcal{X}$
1.1  List CC, SC, Evidence
1.2  char *cat_h, *cat_p, *Set, InconsistentC, InconsistentS
1.3  int x, i, j
1.4  GetInconsistenceC($\Sigma_c$,CC)
1.5  int num_h = total(CC)
1.6  for i ← 1 to num_h do
1.7      ClCategory=GetConflictingCategory(); cat_c=Category(CC[i],1)
1.8      cat_p=Category(CC[i],2)
1.9      InconsistentC=GetInconsistentCC(CC[i])
1.10     x = total(InconsistentC)
1.11     for j ← 1 to x do
1.12         Evidence=SearchEvidence(InconsistentC)
1.13         if Evidence ≠ null then
1.14             Update(Evidence)
1.15         else
1.16             Update(GetFirtsElement(cat_p))
1.17 GetInconsistenceS($\Sigma_s$,$\mathcal{D}$, SC)
1.18 int num_s = total(SC)
1.19 for i ← 1 to num_s do
1.20     ClCategory=GetConflictingCategory(); cat_c=Category(SC[i],1)
1.21     cat_p=Category(SC[i],2)
1.22     InconsistentS=GetInconsistentSC(SC[i])
1.23     x = total(InconsistentS)
1.24     for j ← 1 to x do
1.25         Evidence=SearchEvidence(InconsistentS)
1.26         if Evidence ≠ null then
1.27             Update(Evidence)
1.28         else
1.29             Set=Elements(InconsistentS,cat_c,cat_p,ClCategory)
1.30             Update(Set)
```

and the inconsistent dimension $\mathcal{D}$. This algorithm first handles violations of covering constraints (line 1.4), and then manages the strictness constraints (line 1.17). The algorithm captures in the list CC the covering constraints that are violated by dimension $\mathcal{D}$ (line 1.4). Then, for each violated constraint (line 1.6) the algorithm gets the conflicting category that is related with the current covering constraint (variable $ClCategory$), the category in the left hand side of the constraint, called the child category, which is stored in variable cat_c, and the category in the right hand side of the constraint, called the parent category, which is stored in variable cat_p (lines 1.7-1.8). Then, the inconsistent elements are captured in the list $InconsistentC$ (line 1.9). For each of them, the algorithm searches for the existence of *evidence* that can be use to restore consistency (line 1.12). If there is evidence, then the algorithm takes this knowledge to choose an element in the parent category (line 1.14). On the contrary, when no evidence is found, the algorithm chooses the first element in the parent category of the constraint as the ancestor to the inconsistent element (line 1.15). The latter option can leave the dimension inconsistent with respect to strictness constraints, but this problem is fixed later when dealing with this kind of constraints.

Then, the algorithm handles violations of strictness constraints (line 1.17), and for each of the violated constraints (line 1.19), it gets again the conflicting level, the child and parent categories in the constraint, and the inconsistent elements (line 1.22). Then, for each inconsistent element for a given constraint, it searches for evidence of consistency (line 1.25). If such evidence is found, the algorithm updates the dimension according to the collected knowledge (line 1.27). If not, a new composed element is inserted into the parent category (value in variable cat_p) and the dimension is updated such that the inconsistent element will reach this new composed element (line 1.28) .



(a) Inconsistent Dimension $\mathcal{D}$



(b) Compatible Repair $\mathcal{X}$

Figure 4: Inconsistent dimension and its compatible repair

Proposition 1. Given a dimension $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Elem}_\mathcal{D}, <_\mathcal{D})$ and a set Σ of strictness and covering constraints, its compatible repair $\mathcal{X} = (\mathcal{H}_\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathsf{CElem}_\mathcal{X}, \ll_\mathcal{X})$ with respect to Σ is such that $\mathcal{X}$ satisfies Σ. □

Example 9. Consider the dimension in Figure 2(a) which violates the covering constraint Tournament⇒ Confederation and the strictness constraint Team → Confederation. When dealing with the covering constraint, variable $ClCategory$ stores value Confederation, which in this case is the unique conflicting level related with the covering constraint. Variables cat_c takes value Tournament and variable cat_p takes value Confederation, the list $InconsistentC$ stores element ASC from the category Tournament. Then the algorithm searches for evidence of consistency. In this case, there is evidence that element ASC should reach element AFC in category Confederation, since there is a bottom element AU that reaches ASC and goes to AFC in Confederation via element AS in category Zone. Then, the algorithm inserts the edge (ASC,AFC) in the compatible dimension (line 1.14). Then, the algorithm fixes the violations of the strictness constraint. The list $InconsistentS$ contains the inconsistent element AR that reaches two different parents in the conflicting category Confederation. In the inconsistent dimension, the consistent element CH in category Team reaches element SA in category Zone as the inconsistent element AR, therefore, it is an evidence that the edge (SA, CONM) is correct. On the other hand, AR reaches EC in category Tournament and there is no other consistent element that reaches EC. Therefore, the algorithm deletes the edge (EC, UEFA) and inserts the edge (EC, CONM) to restore consistency (line 1.27). Figure 2(d) shows the compatible repair $\mathcal{X}$ obtained by applying Algorithm 1. In this case, the compatible repair coincides with the minimal repair $\mathcal{D}_2$ for $\mathcal{D}$ in Figure 2(c).

Consider now the inconsistent dimension $\mathcal{D}$ in Figure 4(a) for the hierarchy schema in Figure 1(a). Dimension $\mathcal{D}$ violates the strictness constraint Team→Confederation through the element t_2 that goes to elements c_2 and c_3 in the Confederation category. In this case there is no evidence of consistency that help to choose a correct element in the parent category. Then, the algorithm creates the composed element $\{c_2, c_3\}$ which is inserted in the Confederation category. Then, the parents of t_2 are related to it. In this case, elements z_2 in the Zone category and o_2 in the Tournament category. The algorithm returns the compatible repair in Figure 4(b). □

Algorithm 1 runs in $O(c{*}max(n_1, \ldots, n_n)) + O(s{*}max(m_1, \ldots, m_n))$, where c is the number of covering constraints violated in dimension $\mathcal{D}$, each n_i in $n_1, \ldots, n_n$ is the number of inconsistent elements for a covering constraint c_i that is violated in $\mathcal{D}$, s is the number of strictness constraint violated in dimension $\mathcal{D}$, and each m_j in $m_1, \ldots, m_n$ is the number of inconsistent elements for a strictness constraint s_i that is violated in $\mathcal{D}$. Then, the algorithm runs in $O(max(O(c * max(n_1, \ldots, n_n)), O(s * max(m_1, \ldots, m_n))))$.

4.1 Answers from the Clean Compatible Repair

If queries are posed over inconsistent dimensions we get incorrect answers, then we can use the compatible repair to obtain approximate answers to the aggregate queries. The answers can be computed as shown before for extended dimensions.

Example 10. Consider the inconsistent dimension if Figure 2(a), and the fact table $\mathtt{Sales}_1$ that stores the incomes for concept of sales (in US dollars) for national teams:

Sales$_1$		
Team	Date	Total
CH	01-01-2013	6000
AR	01-05-2013	3000
AU	01-08-2013	5000
CH	01-12-2013	8000
AR	01-16-2013	8000
AU	01-20-2013	8000

Sales$_2$		
Team	Date	Total
t_1	01-01-2013	5000
t_2	01-05-2013	4000
t_1	01-12-2013	8000
t_2	01-16-2013	2000

The following aggregate query $\mathcal{Q}$ obtain the total of incomes grouped by confederation: `SELECT` R.Confederation, `SUM`(S.Total) `FROM` R(Team, Confederation) R, Sales S `WHERE` R.Team = S.Team `GROUP BY` R_2.Confederation.

The answer to this query in the inconsistent dimension is $\{$(CONM, 25000), (UEFA, 11000), (AFC,13000)$\}$ which is incorrect since the sales of element AR are considered twice (for element CONM and for element UEFA). The answer obtained in the compatible repair (Figure 2(d)) is $\{$(CONM, 25000), (AFC,13000)$\}$. There is no double counting in this answer since the compatible repair is consistent with respect to the constraints.

Now consider the fact table $\mathtt{Sales}_2$ for elements of the inconsistent dimension in Figure 4(a). The answer to query $\mathcal{Q}$ in the inconsistent dimension is $\{(c_1,\ 13000), (c_2,\ 6000), (c_3,\ 6000)\}$ which is incorrect since the incomes to t_2 are counting twice. The answer in the compatible repair in Figure 4(b), seen as a non-extended dimension is $\{(c_1, 13000), (\{c_2,c_3\}, 6000)\}$, where $(\{c_2,c_3\}, 6000)$ intuitively represents that value 6000 belongs either to c_2 or to c_3. Thus, the answer to the compatible repair, which is an extended dimension, would be $\{(c_1,\ 13000), (c_2,\ [0, 6000]), (c_3,\ [0, 6000])\}$. □

5. CONCLUSIONS

In this paper we study the problem of inconsistent dimensions and propose a technique for cleaning them based on the use of *extended dimensions* which allow for the insertion of unions of elements that already belong to a category. This new extended dimension allow us to express imprecision on the data. Uncertainty and imprecision have been analysed in the context of multidimensional models in [27, 4]. In particular in [4] imprecision is allowed in dimensions values, and uncertainty in measures values in the fact tables.

The extended dimension is of interest by itself since it can be used to both represent and query uncertain data. In order to compute the answers possed to extended dimensions it is possible to post-process the answers obtained from current DWs implementations and therefore getting approximate answers efficiently.

The compatible repair is an extended dimension obtained from the inconsistent dimension that satisfies the strictness and covering constraints. This repair allows to deal with ambiguities that arise from the inconsistencies and can provide ranges for the answers to aggregate queries with functions such as SUM and COUNT. The definition of the canonical repair is inspired by the work proposed by Pedersen et. al. [26], where strictness is restored by inserting new elements in artificial categories. Here we create composed elements to restore strictness but as a last option, also we do not create new categories in dimensions. The compatible repair we present differs from the canonical repair presented in [3] which is a new dimension that represents all the minimal repairs of an inconsistent dimension and therefore cannot be computed in polynomial time.

The problem of repairing relational databases with respect to a set of integrity constraints such that functional dependencies and inclusion dependencies has been extensively studied (cf. [1] for a survey). In [5] it was shown that even though there are several ways to represent a data warehouse using relational models (e.g. star schema or snowflake schema [7]) it is not possible to use the relational repair techniques to compute DW's repairs.

The compatible repair can be considered in the category of data cleaning based in data cleaning [2, 12, 11]. In [8] a framework to improve data quality with respect to conditional functional dependencies is presented. There, database repairs are obtained by attribute modifications and they implement an statistical method that modifies attribute values and ensures that the repairs are accurate.

6. REFERENCES

[1] L. Bertossi. Consistent Query Answering in Databases. *ACM Sigmod Record*, 35(2):68–76, 2006.

[2] L. Bertossi and L. Bravo. Generic and declarative approaches to data cleaning: Some recent developments. In S. Sadiq, editor, *Handbook of Data Quality – Research and Practice*. Springer-Verlag Berlin Heidelberg, 2013.

[3] L. Bertossi, L. Bravo, and M. Caniupán. Consistent Query Answering in Data Warehouses. In *AMW'09*, volume 450, 2009.

[4] D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over Uncertain and Imprecise Data. *The VLDB Journal*, 16(1):123–144, 2007.

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

[6] M. Caniupán and A. Vaisman. Repairing Dimension Hierarchies under Inconsistent Reclassification. In *MOREBI'11*, Springer LNCS 6999, pages 75–85, 2011.

[7] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.

[8] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving Data Quality: Consistency and Accuracy. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, pages 315–326, 2007.

[9] G. Dodge and T. Gorman. *Essential Oracle8i Data Warehousing: Designing, Building, and ManagingOracle Data Warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[10] J. Eder and C. Koncilia. Changes of dimension data in temporal data warehouses. In *DaWaK '01*, pages 284–293, 2001.

[11] W. Fan. Extending Dependencies with Conditions for Data Cleaning. In *The 8th IEEE International Conference on Computer and Information Technology*, pages 185 –190, 2008.

[12] W. Fan, F. Fan, X. Jia, and A. Kementsietsidis. Conditional Functional Dependencies for Capturing Data Inconsistencies. *ACM Trans. Database Syst.*, 33:6:1–6:48, 2008.

[13] H. Garcia-Molina, W. Labio, and J. Yang. Expiring Data in a Warehouse. In *VLDB'98*, pages 500–511, 1998.

[14] H. Gupta and I. S. Mumick. Selection of Views to Materialize Under a Maintenance Cost Constraint. In *ICDT'99*, pages 453–470, 1999.

[15] C. Hurtado and C. Gutierrez. *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, chapter Handling Structural Heterogeneity in OLAP. Idea Group, Inc, 2007.

[16] C. Hurtado, C. Gutierrez, and A. Mendelzon. Capturing Summarizability with Integrity Constraints in OLAP. *ACM Transactions on Database Systems*, 30(3):854–886, 2005.

[17] C. Hurtado and A. Mendelzon. Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. In *ICDT'01*, pages 375–389, 2001.

[18] C. Hurtado, A. Mendelzon, and A. Vaisman. Maintaining Data Cubes under Dimension Updates. In *ICDE'99*, pages 346–355, 1999.

[19] C. Hurtado, A. Mendelzon, and A. Vaisman. Updating OLAP Dimensions. In *DOLAP'99*, pages 60–66, 1999.

[20] H.-G. Kang and C.-W. Chung. Exploiting Versions for On-line Data Warehouse Maintenance in MOLAP Servers. In *VLDB'02*, pages 742–753, 2002.

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

[22] C. Letz, E. T. Henn, and G. Vossen. Consistency in Data Warehouse Dimensions. In *IDEAS'02*, pages 224–232, 2002.

[23] E. Malinowski and E. Zimányi. Hierarchies in a Multidimensional Model: from Conceptual Modeling to Logical Representation. *Data Knowl. Eng.*, 59(2):348–377, 2006.

[24] J.-N. Mazón, J. Lechtenbörger, and J. Trujillo. A Survey on Summarizability Issues in Multidimensional Modeling. *Data Knowl. Eng.*, 68(12):1452–1469, 2009.

[25] T. Pedersen and C. Jensen. Multidimensional Data Modeling for Complex Data. In *ICDE'99*, page 336, 1999.

[26] T. Pedersen, C. Jensen, and C. Dyreson. Extending Practical Pre-Aggregation in On-Line Analytical Processing. In *VLDB'99*, pages 663–674, 1999.

[27] T. Pedersen, C. Jensen, and C. Dyreson. Supporting Imprecision in Multidimensional Databases Using Granularities. In *SSDBM'99*, page 90, 1999.

[28] T. Pedersen, C. Jensen, and C. Dyreson. A Foundation for Capturing and Querying Complex Multidimensional Data. *Inf. Syst.*, 26(5):383–423, 2001.

[29] N. Prat, I. Comyn-Wattiau, and J. Akoka. Combining Objects with Rules to Represent Aggregation Knowledge in Data Warehouse and OLAP Systems. *Data Knowl. Eng.*, 70(8):732–752, 2011.

[30] M. Rafanelli and A. Shoshani. STORM: a Statistical Object Representation Model. In *SSDBM'90*, pages 14–29, 1990.

[31] L. Schlesinger and W. Lehner. Extending Data Warehouses by Semiconsistent Views. In *DMDW'02*, pages 43–51, 2002.

[32] D. Theodoratos and M. Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In *DOLAP'00*, pages 1–8, 2000.

[33] Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. Multiple View Consistency for Data Warehousing. In *ICDE'97*, pages 289–300, 1997.