Extended Dimensions for Cleaning and Querying Inconsistent Data Warehouses

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

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

Mónica Caniupán
Universidad del Bio-Bío
Concepción, Chile
mcaniupan@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 summaries. 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 constrains

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), 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). The fact table Sales (Team, Date, Total) stores the incomes for the concept of sales for each team at a specific date. The multi-dimensional 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

---

1FIFA stands for Fédération Internationale de Football Association.

2For the FIFA, Australia is considered as a country of Asia.
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 \( D \) with respect to a set of strictness and covering constraints \( \Sigma \) is a new dimension that satisfies \( \Sigma \) 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 \( D' \) is a minimal repair of a dimension \( D \) 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 \( D \) with respect to a set of constraints \( \Sigma \), 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 \( H \) is a pair \( (\mathcal{C}_H, \mathcal{P}_H) \), where \( (\mathcal{C}_H, \mathcal{P}_H) \) is an acyclic directed graph. Vertices in the set \( \mathcal{C}_H \) are categories and the edges \( \mathcal{P}_H \) represent the child/parent relations between categories. The transitive and reflexive closure of \( \mathcal{P}_H \) is denoted by \( \mathcal{P}_H^* \). The set of categories \( \mathcal{C}_H \) contains a distinguished top category denoted \( \text{All}_H \), which is reachable from every other category in \( \mathcal{C}_H \) and has no outgoing edges. Sometimes, we will use \( c_a \mathcal{P}_H c_b \) instead of \( (c_a, c_b) \in \mathcal{P}_H \). 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 \( D \) is a tuple \( (\mathcal{H}_D, \mathcal{E}_D, \text{Elem}_D) \), where \( \mathcal{H}_D = (\mathcal{C}_D, \mathcal{P}_D) \) is a hierarchy schema; \( \mathcal{E}_D \) is a set of constants, called elements; \( \text{Elem}_D : \mathcal{C}_D \to \mathcal{P}(\mathcal{E}_D)^3 \) is a function that re-
turns the elements that belong to a category; and the relation \( <_{D} \subseteq \mathcal{E}_{D} \times \mathcal{E}_{D} \) represents the child/parent relation between elements of different categories. We denote by \( \prec \) the reflexive and transitive closure of \( <_{D} \). The following conditions hold: (i) \( \forall c \) is the only element in category \( \text{All}_{H} \). (ii) For all \( c_{i}, c_{j} \in \text{All}_{H} \), if \( c_{i} \neq c_{j} \) then \( \text{Elem}_{D}(c_{i}) \cap \text{Elem}_{D}(c_{j}) = \emptyset \). (iii) For all pair of elements \( a, b \in \mathcal{E}_{D} \) if \( a <_{D} b \) then there exists \( c_{i}, c_{j} \in \text{Elem}_{D}(c) \) such that \( c_{i} \prec_{H} c_{j}, a \in \text{Elem}_{D}(c_{i}) \) and \( b \in \text{Elem}_{D}(c_{j}) \). Condition (iii) ensures that the child/parent relation \( (\prec_{D}) \) only connects elements of categories that are connected in the schema. In this definition of dimension we use the function \( \text{Elem}_{D} \) 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 \( H_{D} = (C_{H}, \prec_{H}) \) in Figure 1(b) is formalized as follows: \( C_{H} = \{\text{Team}, \text{Sponsor}, \text{Zone}, \text{Tournament}, \text{Confederation}, \text{All}\} \), and \( \prec_{H} = \{\text{(Team), Sponsor, Team, Zone, (Team, Tournament), (Sponsor, All), (Zone, Confederation), (Tournament, Confederation), (Confederation, All)}\} \).

The dimension \( D_{\text{C}} = (H_{D}, \mathcal{E}_{C}, \text{Elem}_{C}, \prec_{C}) \) given in Figure 1(c) is formalized as follows:

(i) \( \mathcal{E}_{C} = \{\text{all, CONM, UEFA, AFC, CC, GT, SA, WEU, AS, AC, EC, ASC, CH, SP, AU}\} \);

(ii) \( \text{Elem}_{C}(\text{Team}) = \{\text{CH, SP, AU}\} \), \( \text{Elem}_{C}(\text{Sponsor}) = \{\text{CC, GT}\} \), \( \text{Elem}_{C}(\text{Zone}) = \{\text{SA, WEU, AS}\} \), \( \text{Elem}_{C}(\text{Tournament}) = \{\text{AC, EC, ASC}\} \), \( \text{Elem}_{C}(\text{Confederation}) = \{\text{CONM, UEFA, AFC}\} \), \( \text{Elem}_{C}(\text{All}) = \{\text{all}\} \);

(iii) \( \prec_{C} = \{\text{(CH, CC, (CH, GT), (CH, SA), (CH, AC), (SP, GT), (SP, WEU), (SP, EC), (AU, AS), (AC, (CC, GT), (SA, CONM), (WEU, UEFA), (AS, AFC), (AC, CONM), (EC, UEFA), (ASC, AFC), (CONM, all), (UEFA, all), (AFC, all)}\} \).}

For each pair of categories \( c_{i}, c_{j} \in \text{Elem}_{C}, \) such that \( c_{i} \prec_{H} c_{j} \), there is a rollup relation denoted by \( \mathcal{R}_{C}(c_{i}, c_{j}) \) that has the following set of pairs \( \{(a, b) \mid a \in \text{Elem}_{D}(c_{i}), b \in \text{Elem}_{D}(c_{j}) \} \) and \( a <_{D} b \). For example, the rollup relation between categories Team and Zone denoted by \( \mathcal{R}_{C}(\text{Team, Zone}) \) has the pairs: \( \{(\text{CH, SA}), (\text{SP, WEU}), (\text{AU, AS})\} \).

Let \( \mathcal{R}_{D}(c_{i}, c_{j}) \) be a rollup relation, then: (i) \( \mathcal{R}_{D}(c_{i}, c_{j}) \) is strict if \( \mathcal{R}_{D}(c_{i}, c_{j}) \) is a function, i.e., for all elements \( x, y, z \in \mathcal{E}_{D} \), if \( (x, y) \in \mathcal{R}_{D}(c_{i}, c_{j}) \) and \( (x, z) \in \mathcal{R}_{D}(c_{i}, c_{j}) \) then \( y = z \).

(ii) \( \mathcal{R}_{D}(c_{i}, c_{j}) \) is covering if for all elements \( e \in \mathcal{E}_{D} \) such that \( e \in \text{Elem}_{D}(c_{i}) \), there exists an element \( e' \in \mathcal{E}_{D} \) such that \( e' \in \text{Elem}_{D}(c_{j}) \) and \( (e, e') \in \mathcal{R}_{D}(c_{i}, c_{j}) \).

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 computing correct summarizations [17].

Let \( H = (C_{H}, \prec_{H}) \) be a hierarchy schema and let \( D = (H_{D}, \mathcal{E}_{D}, \text{Elem}_{D}, <_{D}) \) be a dimension such that \( H_{D} = H \). (i) A strictness constraint over \( H \) is an expression of the form \( c_{i} \rightarrow c_{j} \) where \( c_{i}, c_{j} \in \text{Elem}_{H} \) and \( c_{i} \prec_{H} c_{j} \). The dimension \( D \) satisfies the strictness constraint \( c_{i} \rightarrow c_{j} \) if and only if the rollup relation \( \mathcal{R}_{D}(c_{i}, c_{j}) \) is strict. (ii) A covering constraint over \( H \) is an expression of the form \( c_{i} \Rightarrow c_{j} \) where \( c_{i}, c_{j} \in \text{Elem}_{H} \) and \( c_{i} \nprec_{H} c_{j} \). The dimension \( D \) satisfies the covering constraint \( c_{i} \Rightarrow c_{j} \) if and only if the rollup relation \( \mathcal{R}_{D}(c_{i}, c_{j}) \) is covering. A dimension \( D \) satisfies \( \Sigma \) if \( D \) satisfies every constraint in \( \Sigma \). Otherwise, the dimension \( D \) is inconsistent with respect to \( \Sigma \) [5].

Example 4. (example 1 continued) The Football_Team dimension \( D_{\text{F}} = (H_{D}, \mathcal{E}_{D}, \text{Elem}_{D}, <_{D}) \) is consistent with respect to the following set of strictness and covering constraints: \( \Sigma = \{\text{Team} \rightarrow \text{Zone, Team} \rightarrow \text{Tournament, Zone} \rightarrow \text{Confederation, Tournament} \rightarrow \text{Confederation, Team} \rightarrow \text{Zone, Team} \rightarrow \text{Tournament, Zone} \rightarrow \text{Confederation, Tournament} \rightarrow \text{Confederation}\} \).

On the other hand, the dimension \( D_{\text{T}} \) in Figure 2(a) is not consistent with respect to \( \Sigma \). In particular, \( D_{\text{T}} \) violates the strictness constraint \( \text{Team} \rightarrow \text{Confederation} \) since the element AR goes to elements CONM and UEFA in category Confederation. Also, \( D_{\text{T}} \) does not satisfy the covering constraint \( \text{Tournament} \rightarrow \text{Confederation} \) 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 \( D = (H_{D}, \mathcal{E}_{D}, \text{Elem}_{D}, <_{D}) \) and \( D' = (H_{D'}, \mathcal{E}_{D'}, \text{Elem}_{D'}, <_{D'}) \), the distance between them, \( \text{dist}(D, D') \), is defined as \( \{(<_{D'} \setminus <_{D}) \cup (<_{D} \setminus <_{D'})\} \). 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 \( D = (H_{D}, \mathcal{E}_{D}, \text{Elem}_{D}, <_{D}) \) be a dimension and \( \Sigma \) be a set of integrity constraints over \( H_{D} \). (i) A repair of \( D \) with respect to \( \Sigma \) is a dimension \( D' = (H_{D'}, \mathcal{E}_{D'}, \text{Elem}_{D'}, <_{D'}) \) such that \( H_{D'} = H_{D}, \mathcal{E}_{D'} = \mathcal{E}_{D'}, \text{Elem}_{D'} = \text{Elem}_{D}, \) and \( D' \) satisfies \( \Sigma \). (ii) A minimal repair of \( D \) with respect to \( \Sigma \) is a repair \( D' \), such that \( \text{dist}(D, D') \) is minimal among all the repairs of \( D \) with respect to \( \Sigma \).

Example 5. Figure 2(b)-(c) shows the two minimal repairs for the inconsistent dimension \( D_{\text{T}} \) in Figure 2(a). Repair \( D'_{1T} \) is obtained by inserting the edge \( (\text{AR, AC}) \), and deleting the edge \( (\text{AR, EC}) \), which change restores consistency of the strictness constraint \( \text{Team} \rightarrow \text{Confederation} \). The edge \( \text{(ASC, AFC)} \) is inserted in \( D'_{1T} \) in

Figure 2: Inconsistent dimension and its minimal repairs
order to satisfy the covering constraint Tournament \Rightarrow Confederation. Repair \mathcal{D}^t_i 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.

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 \(c_i\) that 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 repaired 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 \(X\) is a tuple \((\mathcal{H}_X, \mathcal{E}_X, \text{CElem}_X, \ll_X)\), where \(\mathcal{H}_X = (\mathcal{C}_H_X, \nearrow H_X)\) is a hierarchy schema; \(\mathcal{E}_X\) is a set of constants, called elements; \(\text{CElem}_X : \mathcal{C}_H_X \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{E}_X))\) is a function that, given a category returns a set of sub-categories in \(\mathcal{E}_X\) and the relation \(\ll_X \subseteq \mathcal{P}(\mathcal{E}_X) \times \mathcal{P}(\mathcal{E}_X)\) that corresponds to the child/parent relation between elements of different categories. We denote with \(\ll_X^\prime\) the reflexive and transitive closure of \(\ll_X\). The following conditions hold: (i) all \(\mathcal{E}_X\) is the only element in category \(\forall c_i \in \mathcal{C}_H_X\), (if \(c_i \neq c_j\) then \(\text{CElem}_X(c_i) \cap \text{CElem}_X(c_j) = \emptyset\), (iii) for all pair of elements \(a \in \text{CElem}_X(c_i)\) and \(b \in \text{CElem}_X(c_j)\) if \(a \ll_X b\) then \(c_i \nearrow H_X c_j\), (iv) for each \(c_i \in \mathcal{C}_H_X\) it holds that: (a) \(\emptyset \notin \text{CElem}_X(c_i)\), (b) if \(e \in \text{CElem}_X(c_i)\) then for each element \(e' \in e\) it holds that \(\{e'\} \in \text{CElem}_X(c_i)\), (v) for the bottom category \(c_0 \in \mathcal{C}_H_X\) it holds that for every \(e \in \text{CElem}_X(c_0)\), \(e\) is a singleton.

Condition (iii) ensures that the child/parent relation \(\ll_X\) only connects elements of categories that are connected in the schema. Condition (iv) ensures that every element that is 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 \(\mathcal{H}_X = (\mathcal{C}_H_X, \nearrow H_X)\) be a hierarchy schema and let \((\mathcal{H}_X, \mathcal{E}_X, \text{CElem}_X, \ll_X)\) be an extended dimension for the schema. For each pair of categories \(c_i, c_j \in \mathcal{C}_H_X\) such that \(c_i \nearrow H_X c_j\), there is a rollup relation denoted by \(\mathcal{R}_X(c_i, c_j)\) that has the following set of pairs \(\{(a, b) | a \in \text{CElem}_X(c_i), b \in \text{CElem}_X(c_j)\}\) and \(a \ll_X b\).

![Figure 3: Extended Dimension \(X_{T}\)](image)

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

We say that an extended dimension \(X\) satisfies \(\Sigma\) if \(X\) satisfies every constraint in \(\Sigma\). Otherwise, the extended dimension \(X\) is inconsistent with respect to \(\Sigma\).

Example 6. A possible extended dimension \(X_{T}\) for the hierarchy schema in Figure 1(b) is shown in Figure 3. The rollup relation between categories Team and Zone, \(\mathcal{R}_X(\text{Team}, \text{Zone})\), has the pairs: \(\{(t_1, z_1), (t_2, z_2), (t_3, z_3), (t_4, z_4), (t_5, z_5), (t_6, z_6)\}\). Note that some of the elements in Zone are sets. The extended dimension \(X\) satisfies the set of constraints \(\Sigma = \{\text{Team} \Rightarrow \text{Zone}, \text{Team} \Rightarrow \text{Confederation}, \text{Team} \Rightarrow \text{Confederation}, \text{Team} \Rightarrow \text{Zone}, \text{Team} \Rightarrow \text{Tour-} \text{name}, \text{Zone} \Rightarrow \text{Confederation}, \text{Team} \Rightarrow \text{Confederation}\}\).

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 \(X_{T}\) 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 \(t_1, t_2\) and \(t_1, t_2\).

Definition 2. For an extended dimension \(X = (\mathcal{H}_X, \mathcal{E}_X, \text{CElem}_X, \ll_X, \ll_X^\prime)\), with \(\mathcal{H}_X = (\mathcal{C}_H_X, \nearrow H_X)\), its associated dimension, denoted \(\mathcal{T}(X)\), is a dimension \((\mathcal{H}_D, \mathcal{E}_D, \text{Elem}_D, \ll_D)\) where:

- \(\mathcal{H}_D = \mathcal{H}_X\),
- \(\mathcal{E}_D = \bigcup_{c_i \in \mathcal{C}_H_X} \text{CElem}_X(c_i)\),
- \(\text{Elem}_D = \text{CElem}_X\) with domain in \(\mathcal{E}_D\), and
- \(\ll_D = \ll_X^\prime\) which relates elements in \(\mathcal{E}_D\).

Since for an extended dimension \(X\) we have a dimension \(\mathcal{T}(X)\) 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.
Table 1: Answers to queries over dimensions $T(\mathcal{X}_{\text{D}})$ and $T(\mathcal{X}_{\text{T}})$ and a fact table $\text{Sales}$

<table>
<thead>
<tr>
<th>Confederation</th>
<th>Year</th>
<th>SUM</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(2010,2011)</td>
<td>400</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>2011</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>2012</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(2012,2013)</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>$t_5$</td>
<td>2010</td>
<td>600</td>
<td>1</td>
</tr>
<tr>
<td>$t_6$</td>
<td>(c_1,c_2)</td>
<td>2011</td>
<td>100</td>
</tr>
<tr>
<td>$t_7$</td>
<td>(c_1,c_2)</td>
<td>(2012,2013)</td>
<td>200</td>
</tr>
<tr>
<td>$t_8$</td>
<td>2013</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>$t_9$</td>
<td>(c_1,c_2,c_3)</td>
<td>2010</td>
<td>400</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>(c_1,c_2,c_3)</td>
<td>(2010,2011)</td>
<td>300</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>(c_1,c_2,c_3)</td>
<td>(2012,2013)</td>
<td>300</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>2013</td>
<td>600</td>
<td>1</td>
</tr>
<tr>
<td>$t_{13}$</td>
<td>(c_1,c_2,c_3)</td>
<td>2010</td>
<td>100</td>
</tr>
<tr>
<td>$t_{14}$</td>
<td>2011</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>$t_{15}$</td>
<td>(c_1,c_2,c_3)</td>
<td>(2012,2013)</td>
<td>300</td>
</tr>
<tr>
<td>$t_{16}$</td>
<td>2013</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>$t_{17}$</td>
<td>$c_3$</td>
<td>(2010,2012)</td>
<td>300</td>
</tr>
<tr>
<td>$t_{18}$</td>
<td>$c_3$</td>
<td>2011</td>
<td>300</td>
</tr>
<tr>
<td>$t_{19}$</td>
<td>$c_3$</td>
<td>2012</td>
<td>500</td>
</tr>
<tr>
<td>$t_{20}$</td>
<td>$c_3$</td>
<td>2013</td>
<td>200</td>
</tr>
<tr>
<td>$t_{21}$</td>
<td>$c_4$</td>
<td>2013</td>
<td>300</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Confederation</th>
<th>Year</th>
<th>SUM</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>2010</td>
<td>[0,1700]</td>
<td>0</td>
</tr>
<tr>
<td>$c_1$</td>
<td>2011</td>
<td>[400,1200]</td>
<td>1</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2012</td>
<td>[200,1200]</td>
<td>1</td>
</tr>
<tr>
<td>$c_1$</td>
<td>2013</td>
<td>[0,1700]</td>
<td>0</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2010</td>
<td>0,1400</td>
<td>0</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2011</td>
<td>[600]</td>
<td>0</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2012</td>
<td>[0,1000]</td>
<td>0</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2013</td>
<td>0,2200</td>
<td>0</td>
</tr>
<tr>
<td>$c_3$</td>
<td>2010</td>
<td>0,1100</td>
<td>0</td>
</tr>
<tr>
<td>$c_3$</td>
<td>2011</td>
<td>[300,800]</td>
<td>1</td>
</tr>
<tr>
<td>$c_3$</td>
<td>2012</td>
<td>[500,1600]</td>
<td>1</td>
</tr>
<tr>
<td>$c_4$</td>
<td>2013</td>
<td>[200,2100]</td>
<td>1</td>
</tr>
<tr>
<td>$c_4$</td>
<td>2013</td>
<td>[300,300]</td>
<td>1</td>
</tr>
</tbody>
</table>

Since $Q((T(\mathcal{X}_1), \ldots, T(\mathcal{X}_i), \ldots, T(\mathcal{X}_n)), F)$ 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 $Q$, an extended dimension $\mathcal{X} = (\mathcal{H}_X, \mathcal{E}_X, \mathcal{CElem}_X, \mathcal{CElem}_X, \mathcal{CElem}_X, \mathcal{CElem}_X)$, a fact table $F$, and a tuple $t = (e_1, \ldots, e_n)$ with $e_i \in \mathcal{E}_X$ for every $i \in [1, n]$, the set of tuples associated to $t$ in $Q((T(\mathcal{X}_1), \ldots, T(\mathcal{X}_i), \ldots, T(\mathcal{X}_n)), F)$, denoted by $A(t, Q, \{X_1, \ldots, X_i, \ldots, X_n\}, F)$, is $\{a|a = (s_1, \ldots, s_n, v), a \in Q(T(\mathcal{X}_1), \ldots, T(\mathcal{X}_i), \ldots, T(\mathcal{X}_n)), F), e_i \in s_i \text{ for } i \in [1, n]\}$.

Intuitively, $A(t, Q, \{X_1, \ldots, X_i, \ldots, X_n\}, F)$ contains all the tuples that might contribute to answer tuple $t$.

Definition 5. Given a query $Q$ with aggregate function $\text{SUM}$ or $\text{COUNT}$, a set of extended dimension $\mathcal{S} = \{X_1, \ldots, X_i, \ldots, X_n\}$ where $X_i = (\mathcal{H}_X, \mathcal{E}_X, \mathcal{CElem}_X, \mathcal{CElem}_X, \mathcal{CElem}_X, \mathcal{CElem}_X)$, and a fact table $F$, the answers of $Q$ over $X$ and $F$ is denoted by $Q(S, F)$. A tuple $(e_1, \ldots, e_n, [a, b])$ with $e_i \in \mathcal{E}_X$ for every $i \in [1, n]$ and $[a, b]$ a numerical range belongs to $Q(S, F)$ if and only if: (i) $A((e_1, \ldots, e_n), Q, S, F) \neq \emptyset$; (ii) one of the following holds: either $a = v$ when there exists $v$ such that $\{(e_1, \ldots, e_n), v\} \in Q((T(\mathcal{X}_1), \ldots, T(\mathcal{X}_i), \ldots, T(\mathcal{X}_n)), F)$ or $a = 0$ when there does not exist such $v$; and $b = \sum(s_1, \ldots, s_n, v) \in v$ where $V = A((e_1, \ldots, e_n), Q, S, F)$.

If we consider, instead of $\text{SUM}$ and $\text{COUNT}$ 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 $\text{MIN}$ and $\text{MAX}$, 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 $\text{SUM}$ and $\text{COUNT}$ over the extended dimensions $\mathcal{X}_{\text{T}}$ and $\mathcal{X}_{\text{TIME}}$ in Figure 3, which are obtained considering the answers given in Table 1.
where 1700 is the sum of the aggregate values for tuples in the set $A(t, \{X_2, X_{time}\}, Sales)$. For tuple $t = (c_1, 2011)$ set $A(t, \{X_2, X_{time}\}, Sales)$ 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 $(c_1, 2011, [400, 1200])$ 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 $A(t, \{X_2, X_{time}\}, Sales)$. For the tuple $t = (c_4, 2010)$ set $A(t, \{X_2, X_{time}\}, Sales)$ 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 $t = (c_6, 2013)$ the set $A(t, \{X_2, X_{time}\}, Sales)$ has the unique tuple $t_21$ in Table 1, then, the final answer for $SUM$ is $(c_6, 2013, [300, 300])$ where value 300 is the answer for $t = (c_6, 2013)$ 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 by the unique repair, to be used to compute approximate answers to aggregate queries. The definition of the compatible repair is inspired in [19]: Given an extended dimension $X$ is a dimension that might have set of elements in conflicting 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 $(H_X, \mathcal{E}_X, \mathcal{G}_\text{Elem}_X, \ll, x)$ over a schema $H_X = (C_{H_X}, \mathcal{P}_{H_X})$, a pair of categories $c_i$ and $c_j$ such that $c_j \not\ll_X c_i$, a pair of elements $a, b$ with $a \in \mathcal{G}_\text{Elem}_X(c_i)$ and $b \in \mathcal{G}_\text{Elem}_X(c_j)$. A category $c_k$ such that $c_j \not\ll_X c_k$ is a conflicting category, if there exists a category $c_m$ such that $c_m \not\ll_X c_i$, there is an alternative path between $c_m$ and $c_k$ not including the edge $(c_i, c_j)$, 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 $c_i = Zone$, $c_2 = Team$, Confederation, and $c_m = Team, Tournament$ (Team, Tournament, Confederation) form an alternative path from Team to Confederation not including the edge (Zone, Confederation).

Definition 6. Given a dimension $D = (H_D, \mathcal{E}_D, \mathcal{G}_\text{Elem}_D, \ll_D$) which is inconsistent with respect to a set $\Sigma$ of strictness and covering constraints. Dimension $X' = (H_X, \mathcal{E}_X, \mathcal{G}_\text{Elem}_X, \ll_X)$ is the compatible repair of $D$ with respect to $\Sigma$ 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 $\Sigma$ of strictness and covering constraints, and the inconsistent dimension $D$. 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 $D$ (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 $CICcategory$), 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 used 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 operation 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).

---

**Algorithm 1:** Algorithm to Compute the Compatible Repair

**Input:** $\Sigma$, $D$

**Output:** Compatible Repair $X'$

1.1 List CC, SC, Evidence
1.2 char *cat_h, *cat_p, *Set, InconsistentC, InconsistentS
1.3 int x, i, j
1.4 GetInconsistency($\Sigma_x$, CC)
1.5 int num_h = total(CC)
1.6 for $i = 1$ to $num_h$
1.7 $CICcategory$ = 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 $x$
1.12 Evidence = SearchEvidence(InconsistentC)
1.13 if Evidence $\neq$ null then
1.14 Update(Evidence)
1.15 else
1.16 Update(GetFirstElement(cat_p))
1.17 GetInconsistencyS($\Sigma_x$, $D$, SC)
1.18 int num_s = total(SC)
1.19 for $i = 1$ to $num_s$
1.20 $CICcategory$ = GetConflictingCategory(); $cat_c$ = Category(SC[i][1])
1.21 $cat_p$ = Category(SC[i][2])
1.22 InconsistentS = GetInconsistentCC(SC[i])
1.23 $x = total(InconsistentS)$
1.24 for $j = 1$ to $x$
1.25 Evidence = SearchEvidence(InconsistentS)
1.26 if Evidence $\neq$ null then
1.27 Update(Evidence)
1.28 else
1.29 SetElements(InconsistentS, cat_c, cat_p, $CICcategory$)
1.30 Update(SC)}}
Figure 4: Inconsistent dimension and its compatible repair

Proposition 1. Given a dimension \( D = (H_D, \mathcal{E}_D, \text{Elem}_D, <_p) \) and a set \( \Sigma \) of strictness and covering constraints, its compatible repair \( \mathcal{X} = (H_X, \mathcal{E}_X, \text{Elem}_X, <_X) \) with respect to \( \Sigma \) is such that \( \mathcal{X} \) satisfies \( \Sigma \). □

Example 9. Consider the dimension in Figure 2(a) which violates the covering constraint Tournament \( \rightarrow \) Confederation and the strictness constraint Team \( \rightarrow \) Confederation. When dealing with the covering constraint, variable \( CI\text{Category} \) 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 \( Inconsistent\_C \) 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 \( CH \) 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 \( Inconsistent\_S \) 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 \( D_2 \) for \( D \) in Figure 2(c).

Consider now the inconsistent dimension \( D \) in Figure 4(a) for the hierarchy schema in Figure 1(a). Dimension \( D \) violates the strictness constraint Team \( \rightarrow \) 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 \( c_2 \) in the Zone category and \( c_3 \) in the Tournament category. The algorithm returns the compatible repair in Figure 4(b). □

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

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 below for extended dimensions.

Example 10. Consider the inconsistent dimension in Figure 2(a), and the fact table \( Sales_1 \) that stores the incomes for concept of sales (in US dollars) for national teams:

\[
\begin{array}{c|c|c}
\text{Team} & \text{Date} & \text{Total} \\
\hline
\text{CH} & 01-01-2013 & 6000 \\
\text{AR} & 01-05-2013 & 5000 \\
\text{AU} & 01-06-2013 & 5000 \\
\text{CH} & 01-12-2013 & 8000 \\
\text{AR} & 01-16-2013 & 8000 \\
\text{AU} & 01-20-2013 & 8000 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Team} & \text{Date} & \text{Total} \\
\hline
\text{CH} & 01-01-2013 & 5000 \\
\text{AR} & 01-05-2013 & 4000 \\
\text{AU} & 01-12-2013 & 8000 \\
\text{CH} & 01-16-2013 & 2000 \\
\end{array}
\]

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

The answer to this query in the inconsistent dimension is \( \{(\text{CONM}, 25000), (\text{UEFA}, 11000), (\text{AFC}, 13000)\} \) 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 \( \{(\text{CONM}, 25000), (\text{AFC}, 13000)\} \). There is no double counting in this answer since the compatible repair is consistent with respect to the constraints.

Now consider the fact table \( Sales_2 \) for elements of the inconsistent dimension in Figure 4(a). The answer to query \( Q \) 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 get 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.
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 a statistical method that modifies attribute values and ensures that the repairs are accurate.

6. REFERENCES