Consistent Query Answering in Data Warehouses

Leopoldo Bertossi[1], **Loreto Bravo**[2] and **Mónica Caniupán**[3]

[1] Carleton University, Canada
[2] Universidad de Concepción, Chile
[3] Universidad de Bio-Bio, Chile

Abstract. A Data Warehouse (DW) is a data repository that organizes and physically integrates data from multiple sources under special kinds of schemas. A DW is composed by a set of dimensions that reflect the way the data is structured, and the facts that correspond to quantitative data related with the dimensions. A dimension schema is a hierarchical graph of categories. A dimension instance is *strict* if every element of the dimension has a unique ancestor element in each of the ancestor categories. This property is crucial for the efficiency of the system since it allows for the correct computation of aggregate queries using pre-computed views. A dimension instance may become non-strict after update operations. When this happens, the instance can be *minimally* repaired in several ways. In this paper we characterize consistent answers to aggregate queries by means of smallest ranges that contain the answers obtained from every *minimal* repair. We also introduce the notion of *canonical dimension* which captures information about all the minimal repairs. We use this dimension to approximate consistent query answers.

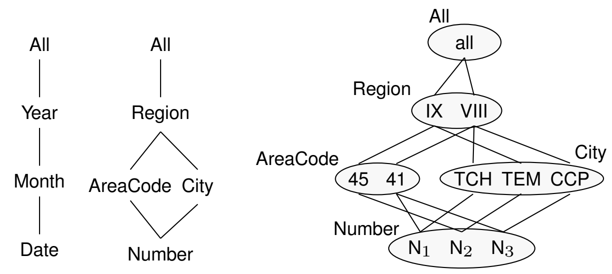
1 Introduction

Data Warehouses (DWs), or more generally, *multidimensional databases*, are data repositories that integrate data from different sources, and keep historical data for analysis and decision support [7]. DWs represent data according to *dimensions* and *facts*. The former reflect the perspectives from which data are viewed, and we may have several of them. The latter corresponds to data (also known as measures) which are generally quantitative and are associated to the different dimensions.

Facts can be aggregated, filtered and referenced using the dimensions. As an illustration, the facts related to the sales of a company may be associated to the dimensions time and location, and should be understood as the sales at certain locations in certain periods of time. A *dimension schema* is usually a hierarchical lattice of category names. A *dimension instance* for the schema assigns sets or extensions to the category names, and also imposes a lattice like structure between elements of different categories.

As an example, the dimension time could be represented by the schema: date→month →year. The multidimensional structure of a DW allows users to formulate aggregate queries at different levels of granularity.

Example 1. A company that manages an online Chilean phone call repository created a *Phone Traffic DW*, with dimensions Time and Phone with the schema shown in Figure 1(a). In the Time dimension, each Date is associated to a Month and each month is associated to a Year which is associated to a category All. On the other hand, in the Phone dimension, each Number is associated to an AreaCode and to a City. Both AreaCode and City are connected to a Region. The top category is All.



Calls			
Number	Date	In	Out
N_1	Jan 1,07	3	0
N_2	Jan 1,07	2	1
N_3	Jan 1,07	5	5
N_1	Jan 2,07	8	0
N_2	Jan 2,07	0	3
N_3	Jan 2,07	5	1

(a) Dimensions schemas (b) Phone dimension instance (c) Fact table

Fig. 1. Phone Traffic DW

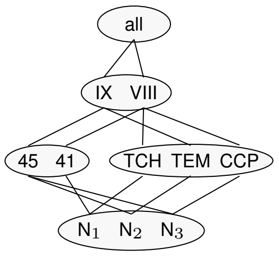
Figure 1(b) shows a dimension instance for the Phone schema. In this instance, TCH (Talcahuano), TEM (Temuco) and CCP (Concepcion) are elements of category City, and IX and VIII are elements of Region. The facts stored in the Phone Traffic DW correspond to number of incoming and outgoing calls of a phone number at a given date, e.g. number N_1 received three calls and made no calls on January 1, 2007. The fact table is shown in Figure 1(c). With this DW we can easily answers queries such as: How many outgoing calls were there on May 2007 per city? How many outgoing calls are there per region every year? □

Generally, a dimension (instance) is required to be *strict*, this is, every element of a category should reach no more that one element in each ancestor category [17, 13]. For example, in the Phone Traffic DW, we expect this property to hold since each number should be associated with a unique city, region and area code. If a dimension instance satisfies its strictness constraint, we say that it is *consistent*. If dimensions are stored as relational tables, strictness can be imposed by a set of functional dependencies over the tables, as the following example shows.

Example 2. The dimension instance in Figure 1 (b) is strict, since, as expected, every number rolls-up to a unique area code, city and region. On the other hand, dimension in Figure 2 is not strict since now element N_3 rolls-up to both IX and VIII in category Region which shows that the data is not accurate and also implies that pre-computed answers at the level of AreaCode and City cannot be used to compute answers for Region. The relational table in Figure 2 maps the edges between the elements of categories Number and Region. Since the functional dependency of Region upon Number does not hold in that table, the dimension instance is not strict. □

Dimensions that are strict and homogenous (cf. Section 2) allow for the correct use of pre-computed answers at low level categories to compute aggregate queries at higher levels. Thus, query answering over strict dimensions increases efficiency of DWs [22]. Even though strictness is important, DWs do not enforce it, and a dimension might become non-strict after an update performed to adapt to changes in data sources or modifications to the business rules [15, 14, 20]. In an enterprise DW, with possibly terabytes of data [7], ensuring strictness of dimensions may be vital for efficient query answering and keeping the data clean.

In [6] the concept of *minimal repair* is formalized, as a strict dimension instance that minimally differs from the given one by a minimum number of changes. Also logic programs to specify and compute them are provided. In that work, the focus is on how



$\mathcal{R}^{\text{Region}}_{\text{Number}}$	
N_1	VIII
N_2	IX
N_3	IX
N_3	VIII

Fig. 2. Non-strict dimension instance (category names are omitted)

to aid the developer or administrator to restore consistency by finding a single repair. However, there might be several alternative repairs, and it is not always possible to know which one is the desirable one.

Here we focus on answering aggregate queries that involve inconsistent dimension instances. In order to obtain semantically meaningful answers, we use the class of all minimal repairs to provide a minimal numerical range to which the answer to the query should belong. With this kind of answer we capture information that is shared by, or invariant under, the minimal repairs of the original instance.

This is a form of consistent query answering (CQA) [3], a problem that has been investigated in the relational setting (cf. [5, 8] for recent surveys). In this paper, a consistent answer to an aggregate query captures through an interval the answers to the same query that would be obtained from each of the minimal repairs if they were materialized. We also introduce the notion of a *canonical dimension instance* for the original, possibly inconsistent, dimension instance, and we show how to obtain it. This dimension captures information about all the minimal repairs, and can be used to approximate the consistent query answers.

The rest of the paper is organized as follows: Section 2 presents the multidimensional model. Next, Section 3 defines repairs of dimension instances and consistent query answer to aggregate queries. Section 4 presents the canonical dimension instance. Related work and some conclusions are discussed in Section 5. This paper presents some initial and ongoing research on dealing with inconsistent DW dimensions.

2 The Multidimensional Model

In this section we present the multidimensional database model that we use as the basic framework for our research. It is described in detail in [13]. A dimension schema $\mathcal{S}$ consists of a pair $(\mathcal{C}, \nearrow)$, where $\mathcal{C}$ is a set of *categories*, and $\nearrow$ is a child/parent relation between categories. The dimension schema can be also represented with a directed acyclic graph where the vertices correspond to the categories and the edges to the child/parent relation. The transitive and reflexive closure of $\nearrow$ is denoted by $\nearrow^*$. There are no shortcuts in the schemas, this is, if $\mathsf{C}_i \nearrow \mathsf{C}_j$ there is no category C_k such that $\mathsf{C}_i \nearrow^* \mathsf{C}_k$ and $\mathsf{C}_k \nearrow^* \mathsf{C}_j$. Every dimension schema contains a distinguished top category called All which is reachable from all other categories, i.e. for every $\mathsf{C} \in \mathcal{C}$, $\mathsf{C} \nearrow^* \mathsf{All}$. The leaf categories are called bottom categories. To simplify the presentation and without loss of generality, we assume categories do not have attributes, and schemas have a unique bottom category.

Example 3. The Phone dimension schema $\mathcal{S} = (\mathcal{C}, \nearrow)$ of Figure 1(a) is defined by: $\mathcal{C}$=$\{$Number, AreaCode, City, Region, All$\}$, $\nearrow = \{$(Number, AreaCode), (Number, City), (AreaCode, Region), (City, Region), (Region, All)$\}$. The relation $\nearrow^*$ is $\nearrow \cup \{$(Number, Number), (Number, Region), (Number, All), $\ldots\}$. The bottom category is Number, and its ancestors are AreaCode, City, Region, and All. $\square$

A dimension (instance) $\mathcal{D}$ over a dimension schema $\mathcal{S} = (\mathcal{C}, \nearrow)$ is a tuple $(\mathcal{M}, <)$, such that: (i) $\mathcal{M}$ is a finite collection of ground atoms of the form $\mathsf{C}(\mathsf{a})$ where $\mathsf{C} \in \mathcal{C}$ and a is a constant. If $\mathsf{C}(\mathsf{a}) \in \mathcal{M}$, a is said to be an element of C. The constant all is the only element of category All. Categories are assumed to be disjoint, i.e. if $C_i(a), C_j(a) \in \mathcal{M}$ then $i = j$. There is a function δ that maps elements to categories so that $\delta(\mathsf{a}) = \mathsf{C}_i$ iff $\mathsf{C}_i(\mathsf{a}) \in \mathcal{M}$. (ii) The relation $<$ contains the child/parent relationships between elements of different categories, and is compatible with $\nearrow$: If $\mathsf{a} < \mathsf{b}$, then $\delta(\mathsf{a}) \nearrow \delta(\mathsf{b})$. We denote with $<^*$ the reflexive and transitive closure of $<$.

The *roll-up* relation between categories C_i and C_j, denoted $\mathcal{R}_{\mathsf{C}_i}^{\mathsf{C}_j}(\mathcal{D})$ consists of the set of pairs $\{(\mathsf{a}, \mathsf{b}) \mid \mathsf{C}_i(\mathsf{a}), \mathsf{C}_j(\mathsf{b}) \in \mathcal{M}$ and $\mathsf{a} <^* \mathsf{b}\}$. When the dimension is clear from the context, we denote the roll-up relation simply by $\mathcal{R}_{\mathsf{C}_i}^{\mathsf{C}_j}$.

A dimension instance $\mathcal{D}$ is said to be *homogeneous* [13] if, for every pair of categories $\mathsf{C}_i \nearrow \mathsf{C}_j$ and element a in C_i, there is an element b in C_j such that $\mathsf{a} < \mathsf{b}$. In what follows we restrict ourselves to homogeneous dimensions. Moreover, a dimension $\mathcal{D} = (\mathcal{M}, <)$ is *strict* if, for every different elements a, b, c with $\mathsf{a} <^* \mathsf{b}$ and $\mathsf{a} <^* \mathsf{c}$, it holds $\delta(\mathsf{b}) \neq \delta(\mathsf{c})$ [12]. In other words, strictness ensures that every relation $<^*$ between two categories is functional. In the paper, we will say that a dimension instance is *inconsistent* if it is not strict.

Example 4. The dimension instance in Figure 1(b) is defined over the Phone dimension schema in Figure 1(a), and consists of:

$\mathcal{M}$ = $\{$Number(N_1), Number(N_2), Number(N_3), AreaCode(45), AreaCode(41), City(TCH), City(TEM), City(CCP), Region(IX), Region(VIII), All(all)$\}$,

$<$ = $\{$(N_1,41), (N_2,45), (N_3,41), (N_1,TCH), (N_2,TEM), (N_3,CCP), (45,IX), (41,VIII), (TCH,VIII), (TEM,IX), (CCP,VIII), (IX,all), (VIII,all)$\}$.

Relation $<^*$ contains all the elements in $<$ plus others, such as (N_1,N_1) and (N_1,VIII).

The dimension instance in Figure 1(b) is both homogeneous and strict. In comparison, the dimension instance in Figure 2 is homogeneous, but not strict since N_3 rolls-up to both IX and VIII in category Region. $\square$

A dimension instance is *sumarizable* if it allows to compute answers to aggregate queries using other pre-computed queries. As an illustration, consider the DW in Figure 1, the query that request the number of calls grouped by Region, can be computed by using pre-computed answers at category AreaCode or City, if any. This will be more efficient since the pre-computed tables will be smaller than the fact tables.

A dimension is summarizable if it is both homogeneous and strict [17].[4] Due to our homogeneity assumption, to ensure summarizability we only need strictness. For instance, the dimension instance in Figure 1(b) is summarizable since it is homogeneous and strict. In contrast, the dimension instance in Figure 2 is not summarizable since it is not strict. If a DW is not summarizable, it will either return incorrect answers if

[4] A requirement for summarizability which is not related to the dimension but to the query is that only distributive aggregate functions should be used (e.g. MAX, MIN, SUM, and COUNT).

using pre-computed views, or it will lose efficiency by needing to compute the answers starting from the bottom category.

3 Repairs and Consistent Query Answering

In this section we first introduce the concept of *minimal repair* [6], and next, we define consistent query answers using minimal repairs as a basis. Intuitively, a minimal repair is a new instance that is strict and is obtained by a minimum number of changes to the original roll-up relation. To compare different repairs, we use the *distance* between the given inconsistent instance and its repairs.

Let $\mathcal{D}=(\mathcal{M}, <_{\mathcal{D}})$ and $\mathcal{D}'=(\mathcal{M}, <_{\mathcal{D}'})$ be dimension instances over the same schema $\mathcal{S}$. The distance between $\mathcal{D}$ and $\mathcal{D}'$ is defined as $dist(\mathcal{D}, \mathcal{D}') = |(<_{\mathcal{D}'} \smallsetminus <_{\mathcal{D}}) \cup (<_{\mathcal{D}} \smallsetminus <_{\mathcal{D}'})|$, i.e. the cardinality of the symmetric difference between the two roll-up relations. Now, we define the notions of repair and minimal repair.

Definition 1. [6] Given a dimension instance $\mathcal{D} = (\mathcal{M}, <)$ over a schema $\mathcal{S}$: (i) a repair of $\mathcal{D}$ is a dimension instance $\mathcal{D}' = (\mathcal{M}', <')$ over $\mathcal{S}$, such that $\mathcal{D}'$ is strict and $\mathcal{M}' = \mathcal{M}$; (ii) a *minimal repair* of $\mathcal{D}$ is a repair $\mathcal{D}'$, such that $dist(\mathcal{D}, \mathcal{D}')$ is minimal among all the repairs of $\mathcal{D}$. (iii) $Rep(\mathcal{D})$ denotes the class of minimal repairs of $\mathcal{D}$. □

Notice that a repair $\mathcal{D}'$ of a dimension $\mathcal{D}$ contains the same elements as $\mathcal{D}$. This restriction is necessary since otherwise a repair could contain less elements in the bottom category, and therefore, data from the fact tables would be lost in the aggregations. On the other hand, repairs do not introduce new elements into a category. These new elements would have no clear meaning, and would not be useful when posing aggregate queries over the categories that contain them. For example, it is not clear what is the meaning of an element λ in a category Month.

Example 5. The dimension instances in Figure 3 are repairs of the non-strict dimension instance $\mathcal{D}$ in Figure 2. All the repairs are obtained from $\mathcal{D}$ by performing insertions and/or deletions of edges. For example, $\mathcal{D}_1$ is generated by deleting edge (CCP,VIII) and inserting (CCP,IX). The distances between the repairs and the original dimension instance $\mathcal{D}$ are: (a) $dist(\mathcal{D}, \mathcal{D}_1) =$ |(CCP,IX), (CCP,VIII)| $= 2$. (b) $dist(\mathcal{D}, \mathcal{D}_2) =$ |(N$_3$,41), (N$_3$,45)| $= 2$. (c) $dist(\mathcal{D}, \mathcal{D}_3) =$ |(N$_3$,TEM), (N$_3$,CCP)| $= 2$. (d) $dist(\mathcal{D}, \mathcal{D}_4) =$ |(45,VIII), (TEM,VIII), (45,IX), (TEM,IX)| $= 4$. Dimensions $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ are minimal repairs since they are closer to $\mathcal{D}$ than $\mathcal{D}_4$. □

Notice that we restore consistency by deleting or inserting edges between elements in directly connected categories. Also, we use a cardinality-based repair semantics instead of the set-inclusion-based [3], which is more common in the relational case. This is because, we assume that inconsistencies arise from a minimal number of errors and thus repairing using cardinality based repairs is more natural. As an illustration, all the repairs in Figure 3 would be minimal repairs under the set inclusion approach, since none of the set differences is a subset of other. In particular dimension $\mathcal{D}_4$ would be a minimal repair even though it is not really a good repair. Indeed, it modifies not only the roll-ups of N$_3$ (which is involved in the inconsistencies), but changes the roll-ups of number N$_2$ which is not even directly involved in the inconsistencies.

As established in [6], there always exists a repair of a dimension $\mathcal{D} = (\mathcal{M}, <)$; and in every minimal repair $\mathcal{D}' = (\mathcal{M}, <')$, it holds $|<'| \leq |<|$. If an instance is already strict, then it is its only minimal repair.

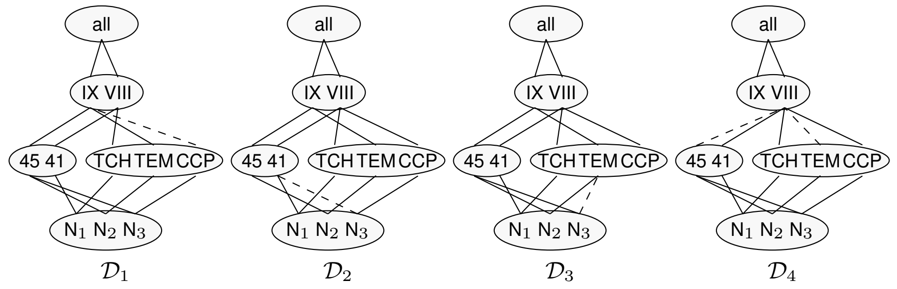


Fig. 3. Repairs of dimension in Figure 2 (dashed edges were inserted to restore strictness)

The most common aggregate queries in DWs are those that perform grouping by the values of a set of attributes, and return a single aggregate value per group:

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

$\mathsf{A}_j, \ldots \mathsf{A}_n$ are attributes of the fact table T or the roll-up functions $\mathsf{R}_i, \ldots \mathsf{R}_m$ (treated as tables), and f is one of min(A), max(A), count(A), sum(A), avg(A), applied to attribute A, with $\mathsf{A} \cap \{\mathsf{A}_j, \ldots \mathsf{A}_n\} = \emptyset$.

Intuitively, a consistent answer to one of those queries will be a range for each group that contains the aggregation values obtained from all the minimal repairs. This definition is based on and extends the notion of consistent answer to a *scalar* (i.e. group-by free) aggregate query presented in [4].

Definition 2. Given a dimension instance $\mathcal{D}$ and an aggregate query $\mathcal{Q}$, a tuple of the form $\langle t_1, \ldots, t_n, [a, b]\rangle$ is a *consistent answer* to query $\mathcal{Q}$ if: (i) $[a, b]$ is a numerical interval; (ii) for every minimal repair $\mathcal{D}'$ of $\mathcal{D}$ tuple $\langle t_1, \ldots, t_m, f(t_1, \ldots, t_n)\rangle$ is an answer to $\mathcal{Q}$ in $\mathcal{D}'$ and $f(t_1, \ldots, t_n) \in [a, b]$; and (iii) there is no smaller interval $[a', b']$ for which condition (ii) holds. □

The non-aggregate portion $\langle t_1, \ldots, t_n\rangle$ of a consistent answer contains the values for the attributes in the `SELECT` clause (which are the same as in the `GROUP BY` clause), and is a consistent answer in the usual, non-aggregate, sense [3]. If the query is *scalar*, we have a single numerical interval, without an associated tuple. The extreme values of the *consistent interval* $[a, b]$ are called, respectively, the *greatest lower bound answer* (glb) and the *least upper bound answer* (lub) to Q for $\langle t_1, \ldots, t_m\rangle$ in $\mathcal{D}$. If $a = b$, then the interval can be represented as $[a]$, or simply a. In particular, if the instance is consistent, the intervals will be all of this form.

Example 6. Consider the non-strict dimension in Figure 2 of the ongoing example, and the following roll-up tables:

```
Q: SELECT R.City, SUM(C.In)
   FROM Calls C, R^City_Number R
   WHERE C.Number = R.Number
         AND C.Out<10
   GROUP BY R.City
```

$\mathcal{R}^{\text{City}}_{\text{Number}}(\mathcal{D}_1)$	
N$_1$	TCH
N$_2$	TEM
N$_3$	CCP

$\mathcal{R}^{\text{City}}_{\text{Number}}(\mathcal{D}_2)$	
N$_1$	TCH
N$_2$	TEM
N$_3$	CCP

$\mathcal{R}^{\text{City}}_{\text{Number}}(\mathcal{D}_3)$	
N$_1$	TCH
N$_2$	TEM
N$_3$	TEM

The answers to the query above are: $\langle$TCH,11$\rangle$, $\langle$TEM,2$\rangle$, $\langle$CCP,10$\rangle$ in repairs $\mathcal{D}_1$ and $\mathcal{D}_2$, and $\langle$TCH,11$\rangle$, $\langle$TEM,12$\rangle$ in $\mathcal{D}_3$. Thus, the consistent answers to Q are $\langle$TCH,11$\rangle$,$\langle$TEM, $[2, 12]\rangle$. City CCP is in the answers from repairs $\mathcal{D}_1$ and $\mathcal{D}_2$ but not from $\mathcal{D}_3$, therefore there is no consistent answer for it. □

A *cuboid* query is an aggregate query where the selections in the `WHERE` *condition* involve only attributes of the fact table and joins involve any attribute. This type of queries are the most common in DWs and correspond to posing an aggregate query in the fact table and then aggregating to a certain level of the dimension. The query in Example 6 is cuboid since the selection condition C.Out<10 refers to an attribute of the fact table. In what follows, we will concentrate in this type of queries.

In [4], it was proved that CQA for scalar aggregate queries under functional dependencies may be intractable. We can also expect intractability in our framework.

Proposition 1. There is an aggregate query with `COUNT` over an attribute such that deciding if the *glb* of the consistent answer is not greater than a given integer is NP-hard.

Proof. We reduce the NP-complete Hitting Set Problem (HSP) $\langle \mathcal{S}, k\rangle$ to our problem. Here $\mathcal{S}$ is a collection $S_1, \ldots, S_m$ of subsets of a base set S, and $k \in \mathbb{N}$. We have to decide if there is a subset $S^{(k)}$ with $|S^{(k)}| \leq k$ and $|S^{(k)} \cap S_i| = 1$, for every i. The schema contains the categories Set, Element, and All with Set $\nearrow$ Element $\nearrow$ All. The following dimension $\mathcal{D}$ is constructed: $\mathcal{M} = \{\mathsf{Set}(i) \mid i = 1, \ldots, m\} \cup \{\mathsf{Element}(x) \mid x \in S\}$, and $<= \{(i,x) \mid x \in S_i\}$. This dimension may not be strict if one of the S_i contains more that one element. In this case, minimal repairs are obtained by edge deletions only. The HSP has a positive solution iff *glb*(`SELECT COUNT`(R.Element) `FROM` $\mathcal{R}_{\mathsf{Set}}^{\mathsf{Element}}$R) $\leq k$. □

4 The Canonical Dimension

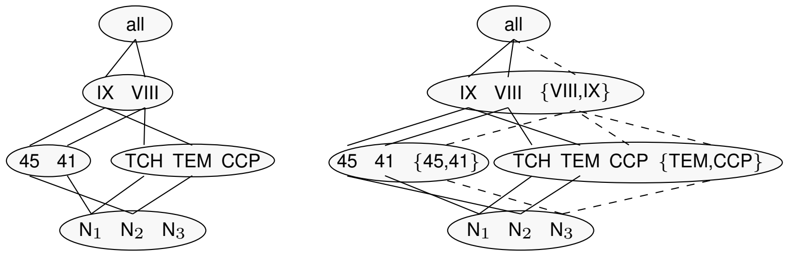
It may be expensive to compute consistent answers by querying all the minimal repairs, which would be a naive approach directly inspired by the definition of consistent answer. A good alternative would be to find a new dimension instance that represents the repairs, in the sense that by querying it, at least an approximation to the consistent answers can be computed. A possible choice could be the *core* dimension, that contains the intersection of the roll-up relations of all the possible minimal repairs.

The core dimension of the ongoing example is shown in Figure 4(a). It does not conform to the original schema, since elements N_3 and CCP do not have any ancestors. Furthermore, the information of number N_3 stored in the fact table will be lost.

An alternative to the core could be a new dimension constructed from the repairs by isolating the elements involved in inconsistencies. For example, if an element a rolls-up to b_1 in a repair, and to b_2 in another, in the canonical dimension, we can add an element $\{\mathsf{b}_1,\mathsf{b}_2\}$ to which element a rolls up to.

Definition 3. Given a dimension instance $\mathcal{D}$ over schema $\mathcal{S}$, the set of *repair parents* in category C for an element a is: $R_{\mathcal{D}}(\mathsf{a},\mathsf{C}) = \{\mathsf{b} \mid \delta(\mathsf{b}) = \mathsf{C}$, and there exists $(\mathcal{M}, <_{\mathcal{D}_i})$ in $Rep(\mathcal{D})$ such that $\mathsf{a} <_{\mathcal{D}_i} \mathsf{b}\}$. □

The repair parents consist of the elements to which element a rolls-up to in category C in some repair of $\mathcal{D}$. If an element a does not roll up to the same element in C in all the repairs, i.e. $|R_{\mathcal{D}}(\mathsf{a},\mathsf{C})| > 1$, the roll-up relation between a and category C is involved in an inconsistency. On the other hand, $\mathcal{D} = (M, <)$ defined over $\mathcal{S} = (\mathcal{C}, \nearrow)$ is consistent iff for every $\mathsf{a} \in \mathcal{M}$ and $\mathsf{C} \in \mathcal{C}$, it holds that $R_{\mathcal{D}}(\mathsf{a},\mathsf{C}) \leq 1$.



(a) Core dimension (b) Canonical dimension

Fig. 4. Core and canonical dimensions for the dimension in Figure 2

Definition 4. Given a dimension instance $\mathcal{D} = (\mathcal{M}, <_{\mathcal{D}})$ over schema $\mathcal{S}$, the *canonical dimension*, denoted $\mathit{Canonical}(\mathcal{D})$, is a dimension instance $(\mathcal{M}', <')$ over $\mathcal{S}$ constructed as follows:

(i) First, set $\mathcal{M}' = \{\mathsf{C}(\{\mathsf{a}\}) \mid \mathsf{C}(\mathsf{a}) \in \mathcal{M}\} \cup \{\mathsf{C}(R_{\mathcal{D}}(\mathsf{a}, \mathsf{C})) \mid \mathsf{C}(\mathsf{a}) \in \mathcal{M}\}$, and
$<' \ = \{(\{\mathsf{a}\}, R_{\mathcal{D}}(\mathsf{a}, \mathsf{C})) \mid \delta(\mathsf{a}) \nearrow \mathsf{C}\}$.

(ii) For each category C, if $\mathsf{C}(\alpha) \in \mathcal{M}'$, then, for every B where $\mathsf{C} \nearrow \mathsf{B}$, let $\beta = \{\mathsf{b} \mid \exists \mathsf{a} \in \alpha, \mathsf{a} <' \mathsf{b}, \mathsf{b} \in B\}$ and $\mathcal{M}' = \mathcal{M} \cup \{\beta\}$ and $<' = <' \cup (\alpha, \beta)$. Repeat this step until no more elements or roll-up relations are added. □

Intuitively, the canonical dimension separates the elements involved in inconsistencies from the ones that are not. The domain of the canonical dimension differs from the one of the original instance, but still conforms to the same schema. Notice also that the bottom categories will always have the same elements as the inconsistent instance, and therefore, the same fact tables can be used.

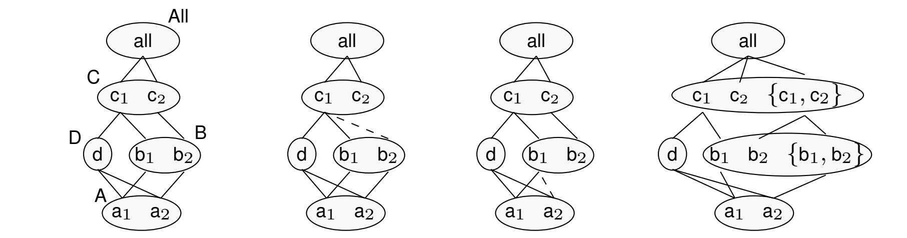
The consistent query answers to a query $\mathcal{Q}$ from $\mathcal{D}$ can be approximated by means of a form of composition of answers to $\mathcal{Q}$ obtained from $\mathit{Canonical}(\mathcal{D})$. We will illustrate the process by means of our running example.

Example 7. Figure 4(b) shows the canonical dimension of the Phone dimension (we omit braces from singletons). In it, N_3 rolls up to element {TEM,CCP} since it rolls up to TEM in repair $\mathcal{D}_3$, and to CCP in repairs $\mathcal{D}_1$ and $\mathcal{D}_2$. The element {TEM,CCP} and the edge (N_3,{TEM,CCP}) are added to the canonical dimension in step (i). The edge ({TEM,CCP}, {VII,IX}) is inserted into the canonical dimension in step (ii).

In the canonical dimension, the roll-up table $\mathcal{R}_{\mathsf{Phone}}^{\mathsf{City}}$ contains {(N_1, TCH), (N_2, TEM), (N_3, {TEM,CCP})}. This is the roll-up table we have to use to answer to query Q in Example 6 from the canonical dimension. In this case, the answers are: ⟨TCH,11⟩, ⟨TEM,2⟩, ⟨{TEM,CCP},10⟩.

Now we will combine these answers as follows. The answers tell us that there were 2 incoming calls to TEM that we are sure of, and 10 calls that could have originated in TEM or in CCP. Thus, we know the number of incoming calls to TEM is in the range $[2, 12]$. In the case of TCH, there is no uncertainty about the number of calls, which are 11. Now, for city CCP there are no incoming calls that we are sure of, therefore, that city is not part of the consistent answers.

In this way we obtain the following composite answers to Q from the canonical dimension: ⟨TCH ,11⟩, ⟨TEM,$[2, 12]$⟩. They can be used to approximate the consistent answers. Actually, in this case, they coincide. □



(a) Inconsistent dimension (b) Repair $\mathcal{D}_1$ (c) Repair $\mathcal{D}_2$ (d) Canonical dimension

Fig. 5. Inconsistent dimension with its repairs and canonical dimension

In this example, since the canonical dimension is strict, the answers obtained from it will be the same independently of the use of pre-computed answers. However, it could be the case that the canonical dimension is not strict. In this case, answers obtained using pre-computed answers may differ from the ones obtained without them, but still will approximate the consistent answers as the following example shows.

Example 8. Consider the inconsistent dimension in Figure 5, the fact table Facts(A,N) = $\{$(a$_1$,1), (a$_2$,2), (a$_2$,2), (a$_1$,2)$\}$ and the cuboid query Q': SELECT R.C, SUM(Facts.N) FROM Facts, $\mathcal{R}_{\mathsf{A}}^{\mathsf{C}}$ R WHERE Facts.A = R.A GROUP BY R.C. The consistent answer to Q' obtained from repairs $\mathcal{D}_1$ and $\mathcal{D}_2$ is $\langle \mathsf{c}_1, 7\rangle$.

In the repairs , element a$_2$ rolls-up to different elements in category B but to the same element in category C. As a result, the canonical dimension (shown in Figure 5(d)) is not strict. The answers obtained from the canonical dimension without using pre-computed answers are $\langle \mathsf{c}_1, 7\rangle$, $\langle \{\mathsf{c}_1\mathsf{c}_2\}, 4\rangle$ which results in the composite answer $\langle \mathsf{c}_1, [7, 11]\rangle$. On the other hand, by using pre-computed answers from category D we get $\langle \mathsf{c}_1, 7\rangle$, which is also the composite answer. By using the pre-computed answers from B we get $\langle \mathsf{c}_1, 3\rangle$, $\langle \{\mathsf{c}_1\mathsf{c}_2\}, 4\rangle$ which results in the composite answer $\langle \mathsf{c}_1, [3, 7]\rangle$.

As it can be observed, all the composite answers can be used as approximation of the consistent answers since they all contain the consistent interval. $\Box$

The following result holds to queries with aggregate functions SUM and COUNT over fact tables with non-negative measures.

Proposition 2. Let $\mathcal{D}$ be a dimension, and $\langle t_1, \ldots, t_n, [a, b]\rangle$ a consistent answer to a cuboid query $\mathcal{Q}$ with aggregation functions SUM or COUNT from $\mathcal{D}$. If $\langle t_1, \ldots, t_n, [c, d]\rangle$ is obtained from the answers to $\mathcal{Q}$ from $\mathit{Canonical}(\mathcal{D})$, then $c \leq a$ and $d \geq b$. $\Box$

The edges between singletons in the canonical dimension correspond to the portion of the inconsistent dimension that is part of all the repairs, and therefore, the values aggregated through them will always be the same or less than the glb of the consistent answer. On the other hand, all the bottom elements that roll-up to an ancestor element c in the inconsistent dimension, will roll-up in the canonical dimension, through all alternative paths of categories in the schema, to an element that contains c. As a consequence, the lub will always be contained in the range obtained from the canonical. Thus, for cuboid queries with SUM or COUNT, the ranges of the consistent answers are contained in those obtained using the canonical dimension.

The consistent answers to queries with aggregate functions MIN and MAX can also be approximated by means of a slightly different composition of the answers obtained from the canonical instance.

It relevant to note, that even though there might be an exponential number of repairs, the size of the canonical dimension is polynomially bounded by the size of the non-strict dimension instance. This is a consequence of two different observations. First, the dimension instance $\mathcal{D}$ and the canonical dimension $Canonical(\mathcal{D})$ have both the same bottom elements. Second, the number of extra elements in a category C of the canonical dimension is at most the summation of the elements in all categories C_i where $\mathsf{C}_i \nearrow \mathsf{C}$.

5 Discussion and Conclusions

In this paper, we analyze CQA in multidimensional data warehouses. We give the notion of consistent answer to an aggregate query with group-by statements. And we also present the canonical dimension that allows us to compute approximate answers. This is part of an ongoing research, and there are still many open problems.

DWs have been conceived as collections of materialized views that extract data from operational databases. Accordingly, much work has been focalized on resolving inconsistencies between operational databases and DWs [10, 11, 23–25, 16]. Only few works have tackled the problem of resolving the inconsistencies in dimensions themselves. This can be due to the fact that early research on DWs considered dimensions as the static part of DWs, being the facts the only part that were affected by updates. Later on, in [15, 14], it was shown that dimensions need to be adapted, due to changes in data sources or the evolution of business rules. When updates affect the DWs, dimensions may become non-strict. Non-strictness may also be caused by inconsistencies between the databases that feed a DW, or by imprecise or erroneous data.

In [18] the authors analyze the importance of enforcing strictness in dimension instances. This is done by imposing constraints on the dimension schema that are used to guide the update operations, with the goal of keeping dimensions strict. In [21] a method to transform non-strict dimensions into strict dimensions is presented. This is done by inserting new artificial elements into categories. As an illustration, if an element a rolls-up to both b and c in the same category, a new element (b,c) is created, and a is associated with this new element. Any other element that was associated to elements b or c becomes now associated to (b,c). Our dimension repairs are not constructed in this way, we restore strictness by inserting or deleting edges between elements, but we do not introduce new elements into categories.

However, we do use the idea of merging elements to define the canonical dimension. This is a unique dimension instance obtained by first isolating the inconsistent data (elements), i.e. those that cause a dimension to be non-strict, and then creating merged elements. These are added into existing categories together with the original ones. In contrast to the method presented in [21], we avoid that consistent data become related with merged data. That is, if an element a_1 rolls up to b, the latter not involved in inconsistencies, it will remain related to b in the repairs, but not to (b,c).

The notion of consistent answer to a first-order query was first defined in [3], in the context of relational databases. CQA for aggregate queries with scalar functions under the range semantics was introduced and analyzed in [4]. The same range semantics was adopted in [1] for scalar aggregate queries in data exchange. CQA for aggregate queries with group-by statements under an extended range semantics was studied in [9], for relational databases and key constraints.

In relational data warehouses, the strictness condition can be captured by means of functional dependencies (FDs). In relational databases, repairs under FDs are always obtained via tuple deletions (or changes of attribute values). Repairs obtained with these techniques could result in dimensions that do not satisfy the dimension schema or where the roll-up tables do not satisfy the transitive property. Another important difference with the classical relational setting is that there, whole tuples are deleted, i.e. database atoms of arity possibly higher than two, whereas in the case of DWs, only binary relationships (edges between elements) are inserted or deleted. Finally, in the case of DWs we use a cardinality-based repair semantics as opposed to the set-inclusion-based, which is more common in the relational case [5]. Repairs that minimize the number of changes result in more reasonable dimensions in the context of DWs. Cardinality-based relational repairs have been studied in detail in [19, 2].

It would also be interesting to provide a more declarative definition of the canonical, and a simpler mechanism to compute it (or what may be relevant of it) from the inconsistent dimension instance, without having to appeal to the explicit minimal repairs. These and other properties of the canonical dimension are subject of ongoing and future research.

Acknowledgements: Leo Bertossi is a Faculty Member of the IBM Center for Advanced Studies (Toronto Lab). Part of this work was done while he was visiting the Universities of Bio-Bio and Concepción. He is very much grateful for the hospitality. Mónica Caniupán is funded by FONDECYT grant #11070186 and Loreto Bravo by FONDECYT grant #11080260 and CONICYT grant PSD-57. We also thank Carlos Hurtado for some useful comments.

References

1. F. Afrati and P. G. Kolaitis. Answering Aggregate Queries in Data Exchange. In *PODS*, pages 129–138, 2008.
2. F. Afrati and P. G. Kolaitis. Repair Checking in Inconsistent Databases: Algorithms and Complexity. In *ICDT*, 2009.
3. M. Arenas, L. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *PODS*, pages 68–79, 1999.
4. M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 296(3):405–434, 2003.
5. L. Bertossi. Consistent Query Answering in Databases. *ACM Sigmod Record*, 35(2):68–76, 2006.
6. M. Caniupan, L. Bravo, and C. Hurtado. Logic Programs for Repairing Inconsistent Dimensions in Data Warehouses. Submitted to Journal Theory and Practice of Logic Programming, Jan 2009.
7. S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.
8. J. Chomicki. Consistent Query Answering: Five Easy Pieces. In *ICDT*, pages 1–17, 2007.
9. A. Fuxman, E. Fazli, and R. J. Miller. Conquer: efficient management of inconsistent databases. In *SIGMOD*, pages 155–166, 2005.
10. H. Garcia-Molina, W. Labio, and J. Yang. Expiring Data in a Warehouse. In *VLDB*, pages 500–511, 1998.
11. H. Gupta and I. S. Mumick. Selection of Views to Materialize Under a Maintenance Cost Constraint. In *ICDT*, pages 453–470, 1999.
12. C. Hurtado and C. Gutirrez. *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, chapter Handling Structural Heterogeneity in OLAP. Idea Group, Inc, 2007.
13. C. A. Hurtado, C. Gutierrez, and A. O. Mendelzon. Capturing Summarizability with Integrity Constraints in OLAP. *ACM Transacations on Database Systems*, 30(3):854–886, 2005.

14. C. A. Hurtado, A. O. Mendelzon, and A. A. Vaisman. Maintaining Data Cubes under Dimension Updates. In *ICDE*, pages 346–355, 1999.
15. C. A. Hurtado, A. O. Mendelzon, and A. A. Vaisman. Updating OLAP Dimensions. In *DOLAP*, pages 60–66, 1999.
16. H. Kang and C. Chung. Exploiting Versions for On-line Data Warehouse Maintenance in MOLAP Servers. In *VLDB*, pages 742–753, 2002.
17. H. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *SSDBM*, pages 132–143, 1997.
18. C. Letz, E. T. Henn, and G. Vossen. Consistency in Data Warehouse Dimensions. In *IDEAS*, pages 224–232, 2002.
19. A. Lopatenko and L. E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, pages 179–193, 2007.
20. A. O. Mendelzon and A. A. Vaisman. Temporal Queries in OLAP. In *VLDB*, pages 242–253, 2000.
21. T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Extending Practical Pre-Aggregation in On-Line Analytical Processing. In *VLDB*, pages 663–674, 1999.
22. M. Rafanelli and A. Shoshani. STORM: a Statistical Object Representation Model. In *SSDBM*, pages 14–29, 1990.
23. L. Schlesinger and W. Lehner. Extending Data Warehouses by Semiconsistent Views. In *DMDW*, pages 43–51, 2002.
24. D. Theodoratos and M. Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In *DOLAP*, pages 1–8, 2000.
25. Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. Multiple View Consistency for Data Warehousing. In *ICDE*, pages 289–300, 1997.