# Handling Inconsistencies in Data Warehouses

Mónica Caniupán

Carleton University,
School of Computer Science,
Ottawa, Canada
mcaniupa@scs.carleton.ca

**Abstract.** Data warehouses (DWs) can become inconsistent when some dimensional constraints are not satisfied by the dimension instances. In this paper, we present preliminary results about the effects of the violation of partitioning constraints in homogeneous dimension instances over aggregation queries, and in particular over the summarizability property (SUMM) of the DWs. We are interested in finding ways to retrieve consistent answers even when the DW is inconsistent. We give a notion of repair for inconsistent instances based on a notion of prioritized minimization. We also describe a notion of consistent answer in DWs.

## 1 Introduction

Data warehouses (DWs) are data repositories that integrate data from different sources and also keep historical data. They can be queried by *OLAP* (On-Line Analytical Processing) systems, which in particular, require aggregation of data stored in the DW [4].

The DWs consist mainly of dimensions and facts. Dimensions reflect the way in which the data is organized. Some some typical dimensions are time, location, customers, etc. The facts correspond to quantitative data related with (a finite number of) dimensions, for example facts related with sales may be associated to the dimensions time, and location, and should be understood as the sales done by the locations in certain periods of time.

DWs can be modelled and implemented by using a relational (ROLAP) or a multidimensional (MOLAP) approach. The multidimensional approach is better than the relational one to support data aggregation, because aggregations can be computed in a straightforward way from the multidimensional structure. We base our work on the *multidimensional model* proposed in [7], where dimensions are modelled by hierarchy schemas together with a set of constraints, while the facts are represented by tables that refer to the dimensions. In this paper we only consider basic dimension schemas, called *strictly homogeneous dimension schemas* (cf. section 2).

Usually, dimensions are considered the static part of the DW, whereas the facts are considered the dynamic part, in the sense that the update operations affect mainly the fact tables. However, in [8, 9] the need to update dimensions is analyzed. They argue that dimensions have to be adapted to changes in data sources or in the business structure. They define a set of update operators for homogeneous dimension schemas and instances.

In the presence of such update operations, DWs may become inconsistent with respect to dimension constraints. We are interested in studying the effects of violations

W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 166–176, 2004.

of a specific class of dimension constraints, the so-called *partitioning constraints* in homogeneous dimension instances (from now on, homogeneous instances) on aggregation queries. These constraints are fundamental for enforcing navigability properties in dimension schemas. One of the effects we will analyze in detail is how the violation of constraints affects the SUMM property of the DWs. The latter is the capability of correctly computing queries (*cube views*) using others pre-computed aggregate views. We will concentrate on queries with aggregation functions, which perform grouping of attributes and return a value for each group.

This analysis has been done on the basis of examples and theoretical work. We will use DB2 data warehousing technology to implement our concepts and mechanisms for CQA.

We also intent to retrieve consistent answers to queries even when the DW is inconsistent. Of course, a characterization of such answers becomes necessary. In order to do this, we use the concept of *repair* of a DW that is inconsistent wrt the dimension constraints. A concept of repair was first introduced in [1] in the context of relational databases and first order integrity constraints. In that framework, a repair is another database instance that minimally differs from the original instance (wrt inclusion of sets of tuples inserted or deleted into/from the original database) and satisfies the given set of integrity constraints. In [2] it is defined the set of consistent answers to aggregation queries with scalar functions (that return a single value for an entire relation). That set is defined as an optimal interval $[a, b]$ such that the evaluation of the query on every repair of the database returns a value $v$ such that $a < v < b$.

We will show that these previous notions of repair are not suitable for the DW framework. In consequence, we give a new definition of repair and consistent answer for multidimensional DWs subject to a set of partitioning constraints and for queries with aggregation functions. DW repairs are used as an auxiliary concept to characterize the consistent answers.

We get dimension instances repairs wrt partitioning constraints by introducing minimal changes over the original inconsistent dimension instances. In order to achieve this, and given that we are considering hierarchical representations with multiple levels, we explore the notion of prioritized minimization (as given in [12]). After that, we give a preliminary definition of consistent answer for such kind of queries. Intuitively, a consistent answer to a query with aggregation function is a set of attributes grouped together with an interval, as defined in [2], for each aggregation function.

For future research we leave the development of a methodology for computing repairs (if necessary, because this should be avoided whenever possible due to its complexity) and consistent answers. In addition, we will extend this study to heterogeneous dimension schemas [7].

## 2 Preliminaries

A hierarchy schema is a directed acyclic graph $(C, \nearrow)$, where $C$ is a set of categories, and $\nearrow$ is a child/parent relation between categories (edges in the graph), $\nearrow^*$ is the transitive and reflexive closure of $\nearrow$. For simplicity, categories do not have any attributes, and there is a distinguished top category named $All$, whose only element is $\{all\}$, that is

reachable from all other categories. The category at the lowest level is named the *bottom* category.

*Example 1.* The figure 1(a) shows the *National Parks* hierarchy schema, with:

- $C = \{$*Park, Type, Location, Country, All*$\}$,
- $\nearrow$ consists of the edges $\{$*(Park, Type), (Park, Location), (Type, Country), (Location, Country),(Country, All)*$\}$; and
- $\nearrow^* = \nearrow \cup \{$*(Park,Park), (Type, Type), (Park, Country), ...*$\}$. □

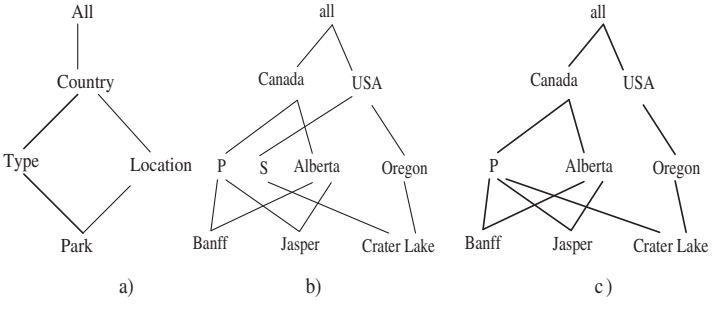


**Fig. 1.** a) Hierarchy schema b),c) Dimension Instances

The hierarchy schema has a domain $D$ that can be infinite, whose elements have a unique name. An instance over a hierarchy schema is defined as a first order logic (FOL) structure of the form 1.

$$\mathcal{D} = \langle D, C_i^D, ..., C_m^D, All^D, all^D, A^D, <^D, <^{*^D} \rangle, \tag{1}$$

where $D$ is the domain of the herbrand structure [13], whose elements $k$ must be interpreted with their own values. $C_i^D, ..., C_m^D, All^D \subseteq D$, are unary predicates that represent categories, and $All^D$ is the top category, with element $\{all^D\}$. $A^D \subseteq \{p_{ij} \mid i, j = 1, ..., m\}$, where each $p_{ij}$ represents an edge between the categories $i$ and $j$ on the hierarchy schema. $<^D \subseteq D \times D$ is the child/parent relation between elements of categories. $<^{*^D} \subseteq D \times D$ is the reflexive and transitive closure of $<^D$. In this sense, $<^{*^D}$ can be seen as an interpreted relation name, which has a fixed interpretation depending on the interpretation of $<^D$. A dimension instance $\mathcal{D}$ indicates relationships between elements of categories which must be connected in the hierarchy schema. This is achieved through the set of the $p_{ij}$, in the sense that: $\mathcal{D} \models p_{ij} \Leftrightarrow p_{ij} \in A^D$.

*Example 2.* $\mathcal{D} = \langle D, Park(\cdot)^D, Type(\cdot)^D, Location(\cdot)^D, Country(\cdot)^D, All(\cdot)^D, all^D, A^D, <^D, <^{*^D} \rangle$ is an instance for the hierarchy schema (a) in Figure 1, where $D$ is composed by names for parks, types, locations, countries, and:

- $Park^D = \{Banff, Jasper, CraterLake\}$, $Type^D = \{P, S\}$,
- $Location^D = \{Alberta, Oregon\}$, $Country^D = \{Canada, USA\}$, $All^D = \{all^D\}$
- $A^D = \{p_{parktype}, p_{parklocation}, p_{typecounty}, p_{locationcountry}, p_{countryAll}\}$
- $<^D$ = {*(Banff, P), (Banff, Alberta), (Jasper, P), (Jasper, Alberta), (CraterLake, S), (CraterLake, Oregon), (P, Canada), (S,USA), (Alberta, Canada), (Oregon, USA), (Canada, All), (USA, All)*}.
  Figure 1(b) illustrates this instance.
- $<^{*^D}$=$<^D \cup$ {*(Banff, Banff), (Banff,Canada)...* } □

A roll-up function is the mapping of the relation $<^*$ between elements of $C_i, C_j$ categories. It is expressed by:

$$\mathcal{R}^{C_j}_{C_i}(\mathcal{D}) := \{(x, y) | C_i(x) \wedge C_j(y) \wedge x <^* y\} \tag{2}$$

This function is fundamental for computing aggregation of data.

Dimension instances must satisfy a set of conditions [7]. The partitioning property is one of them. It is defined by:

$$\forall\, (x, y, z)(C_i(x) \wedge C_j(y) \wedge C_j(z) \wedge x <^* y \wedge x <^* z \rightarrow y = z) \tag{3}$$

It enforces that roll-up functions are functional, and so they allow for the correct computation of aggregations. The SUMM property says that a category $C$ can be computed from other category $C'$ in a dimension instance $\mathcal{D}$ if and only if: $R^C_{Cbottom}(\mathcal{D}) = R^{C'}_{Cbottom}(\mathcal{D}) \bowtie R^C_{C'}(\mathcal{D})$ [7].

We call *specific dimensions constraints* those constraints that model hierarchy schemas [7]. Those constraints are used to specify paths and the existence of distinguished elements in the categories. Homogeneous schemas are those modelled by the *into constraints*, which establish all the paths as mandatory. In those schemas, roll-up functions are expected to be total between elements of categories. A schema is called strictly homogeneous when it has one bottom category.

Finally, we concentrate on queries with aggregation functions of the form 4.

$$\begin{aligned} &SELECT A_1, \ldots A_m, af \\ &FROM T, R_1, \ldots R_m \\ &WHERE\ \{\text{joins conditions and others conditions}\} \\ &GROUP BY A_1, \ldots A_m, \end{aligned} \tag{4}$$

where $A_1, \ldots A_m$ are attributes of the facts table $T$ or of roll-up functions $R_i, \ldots R_m$ (they will be treated as tables), and *af* is one of: `MIN`, `MAX`, `COUNT`,`SUM`, `AVG`, `COUNT`, applied to one attribute $A_i \neq A_1, \ldots A_m$.

## 3 The Need for DW Repairs and Consistent Answers

In general, DWs are conceived as collections of materialized views whose main sources are operational databases. As consequence, much effort has been centered in keeping consistency between the sources and the DW [5, 6, 15, 17]. To the best of our knowledge,

the first work related to consistency in dimension schemas in the sense of [7] is presented in [11]. They argue that a dimension schema is consistent if their instances satisfy the partitioning condition. That notion of consistency is used to guide the update operations on dimension schemas that keep that property satisfied. However, there has been no work so far that tackles the problem of already having an inconsistent dimension instance with respect to a specific class of constraints, but still being able to provide consistent answers, in a sense similar to the notion of consistent answer introduced in [1–3] for relational databases. In this regard, the work presented here is the first attempt to handle the problem of consistent query answering (CQA) in DWs.

The concept of repair was already defined in [1] in the context of relational databases. However, that notion does not capture the minimality required by the natural process of repairing multidimensional DWs. This is the case even if we represent the DW as an instance of a relational DB (the ROLAP approach). We show this with an example.

*Example 3.* Figure 2 shows a snowflake schema [4] for the *National Parks* dimension. "PK" indicates the primary key for each table, and the following first order integrity constraint enforce the partitioning property:

$$IC: \forall xyzwv \;\; Park(x, y, z) \wedge Type(y, w) \wedge Country(w) \wedge \\ Location(z, v) \wedge Country(v) \rightarrow w = v$$

Assume we have the following dimension instance *r*:

- *Park* = {*(Banff, P, Alberta), (Jasper, P, Alberta), (CraterLake, P, Oregon)*},
- *Type* = {*(P, Canada)*},
- *Location* = {*(Alberta, Canada), (Oregon, USA)*},
- *Country* = {*Canada, USA*}.

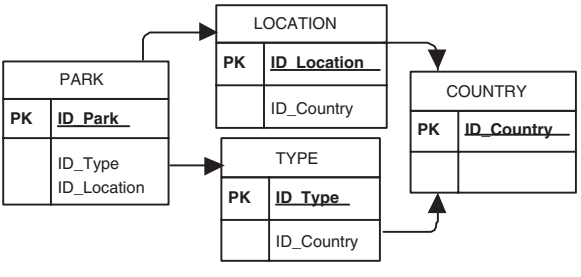


**Fig. 2.** Snowflake schema

Instance *r* does not satisfy the IC because $Canada \neq USA$, and:

$$r \models Park(CraterLake, P, Oregon) \wedge Type(P, Canada) \wedge Country(Canada) \wedge \\ Location(Oregon, USA) \wedge Country(USA)$$

The only possible repair in the relational sense of [1] is obtained by deleting *Park (CraterLake, P, Oregon)*. However, if we apply this change in the multidimensional representation it implies to delete the pairs (*CraterLake, P*) of the roll-up function $R^{Type}_{Park}$ and *(CraterLake,Oregon)* of $R^{Location}_{Park}$, which is not a minimal repair since it is enough to reestablish consistence to delete just one of them. In section 4 we show that to delete both pairs is not a minimal change. □

We will show that good repairs for DWs are obtained by doing minimal deletion of pairs in the roll-up functions involved in the violations of constraints. The problem in the relational model is that a tuple can contain many pairs of those functions (in example 3 a tuple contains two pairs). In that sense, relational model does not allow us to work on a granularity lower than a tuple. Our definition of repair (section 4.1) captures exactly the minimality of changes desired for DWs. We achieve this by identifying the roll-up functions involved in the violations of partitioning constraints, defining a prioritized set of roll-up functions (inspired by the notion of prioritized minimization given in [12]), then those set of functions are manipulated (deletion of pairs) in order to achieve consistence. The repairs are those that reestablish the consistence by doing minimal changes over the prioritized roll-up functions.

On the other side, the importance of the summarizability property of DWs has been analyzed [10, 14]. In [14] a particular class of heterogeneous hierarchies is transformed into homogeneous hierarchies to support summarizability. This is achieved by inserting null values, fusioning other values, and introducing new categories when partitioning constraints are violated. Although these operations, which allow us to get summarizability, could be used for repairing inconsistent DWs, they do not produce minimal repairs. In addition, the fusion of values may produce undesired changes in the semantic of the dimension instances.

## 4 Repairs and Semantically Correct Answers in DWs

Let us show by means of an example how the unsatisfied partitioning constraints (PC) may affect query answering.

*Example 4.* Let $\mathcal{D}$ be the instance in figure 1(c), and $PC : \forall\ (x, y, z)(Park(x) \wedge Country(y) \wedge Country(z) \wedge x <^* y \wedge x <^* z \rightarrow y = z)$. Here, $\mathcal{D} \not\models PC$. As a consequence, the roll-up function: $R^{Country}_{Park} = \{(Banff, Canada), (Jasper, Canada), (CraterLake, Canada), (CraterLake, USA)\}$ is not functional.

Suppose the facts table *Sales*= {(*Banff*, 5000), (*Jasper*, 5000), (*CraterLake*, 10000)} stores sales for national parks, and consider the aggregation query $Q$: "Give the SUM (sales) group by country". The answer for $Q$ is: {(Canada, 20000), (USA, 10000)}. □

Clearly, this result presents an anomaly, the sales of the park *Crater Lake* are added twice, as sales of Canada and also as sales of USA. Now, let us explore how that violation affects the summarization property.

*Example 5.* Consider the following materialized views and roll-up functions:

- *Sales-Type* = {(*P*, 20000)}, *Sales-Loc* = {(*Alberta*, 10000), (*Oregon*, 10000)}
- $R_{Type}^{Country}$ = {(*P, Canada)*}, $R_{Location}^{Country}$ = {(*Alberta, Canada*),(*Oregon, USA*)}

The answers to $Q$: "Give the SUM(sales) group by country" are {(*Canada*, 20000)} and {(*Canada*, 10000), (*USA*, 10000)}, using the respective views and roll-up functions. However, by the summarizability property, the answers must be similar, specially in homogeneous instances, where a category is summarizable from any of the categories below it [7]. □

Given an homogeneous instance $\mathcal{D}$ satisfying the basic properties of the graph structure [7], and a set of partitioning constraints $PC$, we claim:

**Theorem 1.** *$\mathcal{D} \models PC$ if and only if $\mathcal{D} \models SUMM$.* □

This result is important because we can verify summarizability by testing satisfiability of partitioning constraints. This test could be easily performed by using views. In that way, we could identify the elements participating in violations and use that information to fix the dimension instance.

### 4.1 Dimension Instances Repairs

Partitioning constraints can be seen as functional dependencies (FD) in relational databases. The general way to repair inconsistent databases wrt to FDs is by deleting the tuples participating in the violations [2]. However, in dimension instances, there are no tuples in the sense of relational databases, but there exist dimension tuples [7], so we could consider as tuples the pairs in the roll-up functions between elements.

Dimension instances form a hierarchy of roll-up functions. In consequence, we should identify from which roll-up functions pairs are to be deleted in order to get a good repair. Inspired by the notion of prioritized minimization given in [12], we propose to minimize changes, but assigning higher priority to lower categories. For this purpose, we define first levels of categories on a dimension instance, and to each level we associate a set of roll-up functions. Specifically, given a dimension instance $\mathcal{D}$ of the form (1) with maximum distance $n$ among the categories of the graph, we define

**Definition 1.** A level $L_i$ with $0 \leq i \leq n$ is a set of elements belonging to categories with distance $i$ (in the hierarchy schema) to the bottom category. For each level $L_i$ there exists a set $R_i \subseteq <^1$ defined by: $R_i := \{(a, b) | a < b \wedge C_j(a) \in L_i \wedge C_k(b) \in L_{i+1}\}$, where $\{C_j, C_k\}$ are categories of $\mathcal{D}$. □

*Example 6.* For the instances of the figure 3(a):

- $L_0$ = {*Banff, Jasper, CraterLake*},
- $R_0$ = {*(Banff, P), (Banff, Alberta), (Jasper, P), (Jasper, Alberta), (CraterLake, Oregon)*}. □

[1] The child/parent relation among categories elements.

**Definition 2.** The distance $\Delta$ between two dimension instances $\mathcal{D}_1, \mathcal{D}_2$ on the set $R_i$ with $0 \leq i \leq n$ is defined by: $\Delta_i(\mathcal{D}_1, \mathcal{D}_2) = \{(a,b) | (((a,b) \in R_{i,\mathcal{D}_1}) \wedge ((a,b) \notin R_{i,\mathcal{D}_2})) \vee (((a,b) \in R_{i,\mathcal{D}_2}) \wedge ((a,b) \notin R_{i,\mathcal{D}_1}))\}$.

Given a dimension instance $\mathcal{D}$, we define: $\mathcal{D}_1 \leq_{\mathcal{D},i} \mathcal{D}_2 \leftrightarrow \Delta_i(\mathcal{D}, \mathcal{D}_1) \subseteq \Delta_i(\mathcal{D}, \mathcal{D}_2)$. □

*Example 7.* Let $\mathcal{D}$, $\mathcal{D}_1$, $\mathcal{D}_2$ be the instances in figures 1(c), 3(a), and 3(c), respectively. It holds:

- $\Delta_0(\mathcal{D}, \mathcal{D}_1)$ = {*(CraterLake, P)*},
- $\Delta_0(\mathcal{D}, \mathcal{D}_2)$ = {*(CraterLake, P)*, *(CraterLake, Oregon)*},
- $\mathcal{D}_1 \leq_{\mathcal{D},0} \mathcal{D}_2$, because $\Delta_0(\mathcal{D}, \mathcal{D}_1) \subseteq \Delta_0(\mathcal{D}, \mathcal{D}_2)$. □

**Definition 3.** Let $\mathcal{D}, \mathcal{D}_1, \mathcal{D}_2$ be dimension instances over the hierarchy schema $H$ with domain $D$. It holds: $\mathcal{D}_1 \leq_{\mathcal{D}} \mathcal{D}_2$ iff: $\exists i\ ((\Delta_k(\mathcal{D}, \mathcal{D}_1) = \Delta_k(\mathcal{D}, \mathcal{D}_2), k < i) \wedge (\mathcal{D}_1 \leq_{\mathcal{D},i} \mathcal{D}_2))$. □

**Definition 4.** Given a dimension instance $\mathcal{D}$, and a set of partitioning constraints $PC$, a repair of $\mathcal{D}$ wrt $PC$ is a dimension instance $\mathcal{D}'$, such that $\mathcal{D}' \models PC$, and $\mathcal{D}'$ is $\leq_{\mathcal{D}}$-minimal in the class of dimension instances that satisfy $PC$. The set of repairs of $\mathcal{D}$ is denoted by $Repairs_{PC}(\mathcal{D})$. □

*Example 8.* Figures 3(a),(b) show the repairs for the instances in figure 1(c). □

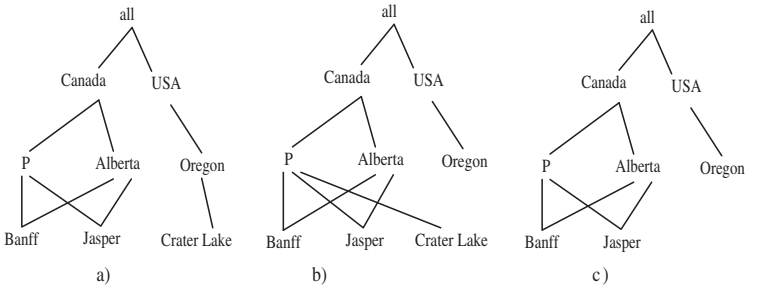


**Fig. 3.** Dimension Instances

### 4.2 Consistent Answers

**Definition 5.** Given a dimension instance $\mathcal{D}$, a set of partitioning constraints $PC$, and a set of repairs $Repairs_{PC}(\mathcal{D})$, the execution of the aggregation query $Q$ over each $\mathcal{D}' \in Repairs_{PC}(\mathcal{D})$ generates a *set of pre-answers* for each $\mathcal{D}'$, $Pre^{Q}_{PC}\mathcal{D}' := \{Q(\mathcal{D}') \mid \mathcal{D}' \in Repairs_{PC}(\mathcal{D})\}$. □

The pre-answers are a set of tuples: $< A_1, \ldots, A_n, Aggr >$, where $A_1, \ldots, A_n$ are attributes of the group-by clause of $Q$, and $Aggr$ is the value for the aggregation function $af$ on $\mathcal{D}'$.

*Example 9.* Consider the repairs in example 8, the sets of pre-answers for $Q$ in example 4 are: {(*Canada*, 10000), (*USA*,10000)} and {(*Canada*,20000)}, respectively. □

**Definition 6.** A consistent answer to an aggregation query $Q$, over a dimension instance $\mathcal{D}$ wrt to a set of partitioning constraints $PC$, is a set of tuples $< A_1, \ldots, A_n, r(Aggr) >$, where $< A_1, \ldots, A_n >$ are the attributes of the group-by clause of $Q$, and $r(Aggr)$ is a range $[a, b]$ of values, $a$ is the greatest-lower-bound (glb), and $b$ is the least-upper-bound (lub) for $Aggr$ in $Pre^{Q}_{PC}(\mathcal{D}')$ for all $\mathcal{D}' \in Repairs_{PC}(\mathcal{D})$. □

*Example 10.* A consistent answer in example 4 is: {(*Canada*, {10000,20000}), (*USA*, {0,10000}). □

## 5 Future Work

A repair is a minimal consistent dimension instance wrt to partitioning constraints. However, a repair is not an homogeneous instance (see figures 3(a), (b)), because some roll-up functions are modified and they are not total anymore. Furthermore, the summarizability property cannot be reestablished in the repairs. However, we could obtain total functions and also summarizability by introducing "dummy" elements in some categories in the repairs, as in [14]. We are analyzing the possible advantages of implementing this idea in the query answering process for DWs. We are also studying the method proposed in [14] to repair DWs and compute consistent answers. We want to see if we get the same consistent answers even when the concept of repair is different.

We could also improve the definition of repairs by using knowledge from equality atoms constraints [7], which impose the existence of certain distinguished elements in the categories. We could require those repairs to satisfy those constraints to get more accurate repairs.

We are developing a methodology for computing consistent answers. We have considered to use the knowledge from partitioning constraints and roll-up functions to compute them, avoiding the computation of all the possible repairs, which is known to be inefficient [1–3].

This is a preliminary study that we will extend to heterogeneous dimension schemas [7], where the situation could be a bit different; mainly because those schemas relax some conditions, becoming more vulnerable to inconsistencies. We already explored update operations over heterogeneous schemas, finding some differences wrt updating homogeneous schemas. Those results will be included in a future publication.

The problem of retrieving consistent answers to aggregation queries is not a new issue, it was already studied for scalar aggregation functions in [2] for relational databases under functional dependencies. We are working on the extension of this work to handle referential constraints in addition to the FDs. We think that future results for this kind

of databases will be useful in the context of DWs, in particular, in the case where *OLAP* operators and DWs are implemented on top of relational databases (ROLAP).

On other side, we are interested in optimizing the query answering process in DWs by taking advantage of aggregation constraints [16]. Also the issue of CQA wrt to aggregation constraints is open. Experiments and implementations will be done on the DB2 platform.

**Acknowledgment.** I am grateful to my PhD supervisor Prof. Leo Bertossi for his support and technical conversations. Research supported by NSERC Discovery Grant 250279-02. We appreciate stimulating conversations with Prof. Alberto Mendelzon and Carlos Hurtado at a very early stage of this research.

## References

1. M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. ACM Symposium on Principles of Database Systems (ACM PODS'99, Philadelphia)*, pages 68–79. ACM Press, 1999.
2. 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, March 2003.
3. P. Barcelo, L. Bertossi, and L. Bravo. Characterizing and computing semantically correct answers from databases with annotated logic and answer sets. *Semantics of Databases (Springer LNCS 2582)*, pages 1–27, 2003.
4. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Rec.*, 26(1):65–74, 1997.
5. H. Garcia-Molina, W. J. Labio, and J. Yang. Expiring data in a warehouse. In *Proc. 24th Int. Conf. on Very Large Data Bases VLDB*, pages 500–511, 1998.
6. H. Gupta and I. S. Mumick. Selection of views to materialize under a maintenance cost constraint. In *Database Theory – ICDT'99: Proc. 7th International Conference, Jerusalem, Israel, January*, Springer LNCS 1540, pages 453–470, 1999.
7. C. Hurtado. *Structurally Heterogeneous OLAP Dimensions*. PhD thesis, Computer Science Dept., University of Toronto, 2002.
8. C. Hurtado, A. Mendelzon, and A.Vaisman. Updating OLAP dimensions. In *Proc. 2nd IEEE-DOLAP Workshop, Kansas City, Missouri, USA*, pages 60–66, 1999.
9. C. Hurtado, A. Mendelzon, and A. Vaisman. Maintaining data cubes under dimension updates. In *Proc. 15th IEEE-ICDE Conference, Sydney, Australia*, pages 346–357, 1999.
10. L. V. S. Lakshmanan, R. T. Ng, C. X. Wang, and Xiaodong. The generalized MDL approach for summarization. In *Proc. 28th Int. Conf. Very Large Data Bases, VLDB, Hong Kong, China, August 20-23*, pages 766–777, 2002.
11. C. Letz, E. T. Henn, and G. Vossen. Consistency in data warehouse dimensions. In *Proc. Int. Database Engineering and Applications Symposium, (IDEAS'02), July 17-19, Edmonton, Canada*, pages 224–232. IEEE Press, 2002.
12. V. Lifschitz. Circumscription. In *Handbook of Logic in AI and Logic Programming, Vol. 3, Oxford University Press*, pages 298–352, 1994.
13. J. W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1984.
14. T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Extending practical pre-aggregation in online analytical processing. In *Proc. 25th Int. Conf. Very Large Data Bases, VLDB, Edinburgh, Scotland*, pages 663–674, 1999.

15. L. Schlesinger and W. Lehner. Extending data warehouses by semi-consistent views. In *Proc. 4th International Workshop of Design and Management of Data Warehouses (DMDW 2002, Toronto, Kanada, May 27), CEUR Workshop Proceedings, Technical University of Aachen (RWTH)*, pages 43–51, 2002.
16. D. Srivastava, K. Ross, P. Stuckey, and S. Sudarshan. Foundations of aggregation constraints. In *PPCP'94: Second Int. Workshop, Orcas Island, Seattle, USA*, Springer LNCS 874, pages 193–204, 1994.
17. D. Theodoratos and M. Bouzeghoub. A general framework for the view selection problem for data warehouse design and evolution. In *Proc. 3rd ACM int. workshop on Data warehousing and OLAP*, pages 1–8. ACM Press, 2000.