Data Warehouse Fixer: Fixing Inconsistencies in Data Warehouses

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

Alvaro Placencia
Universidad del Bío-Bío
Concepción, Chile
aplacenc@ubiobio.cl

***Abstract*—Dimensions in Data Warehouses (DWs) are set of elements connected by a hierarchical relationship. Usually, dimensions are required to be *strict* and *covering* to support summarizations at different levels of granularity. A dimension is strict if all they rollup relations are functions, and is covering if every element in a category is connected with an element in its ancestor categories. We present the *Data Warehouse Fixer* (*DWF*), a system that restores consistency of dimensions that fail to satisfy their strictness constraints. The system checks consistency, computes *minimal repairs* for inconsistent dimensions by implementing Datalog programs with negation and weak constraints, and also fixes inconsistent dimensions.**

***Keywords*-Data Warehouses; inconsistencies; strictness constraints; repairs; repair programs;**

I. Introduction

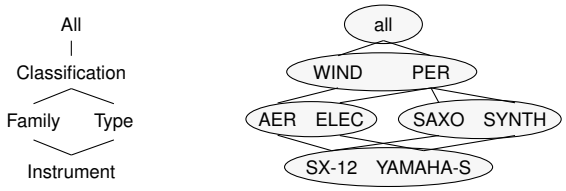
DWs integrate and materialize data from different sources, and keep historical data for decision support [4]. DWs are composed by dimensions and facts. Dimensions reflect the perspectives upon which facts are viewed. Facts (also known as measures) are events that are referenced using the dimension elements. Dimensions are modeled as hierarchies of elements that belong to a specific category. The categories are also organized into a hierarchy called *hierarchy schema*. For each edge in the hierarchy schema, there is a *rollup* relation.

It is usual that DWs use pre-computed summaries at low level categories to derive summaries at higher level categories. In order to use materialized results it is important that dimensions satisfy their *strictness* and *covering* constraints [13], [10], [6]. A *strictness constraint* restricts rollup relations to be functions (many-to-one relations). A *covering constraint* restricts that a rollup relation from category c_1 to category c_2 connect all the elements of c_1 with an element of c_2. A dimension is said to be strict (covering) if all its rollup relations are strict (covering). When dimensions satisfy these conditions we can ensure that facts are aggregated once and no more than once [13], [10], [6].

A. Problem Statement

It has been shown that dimensions need to be updated to adapt them to changes. After updates, dimensions may become inconsistent with respect to their strictness and covering constraints [7], [8].

Example 1: Consider an store that sells musical instruments and has a DW to obtain summarized information. The DW contains a dimension Instrument, among others, with the following hierarchy schema: a *bottom* category Instrument that goes to categories Family and Type. Both Family and Type go to category Classification, which reaches the top category All. Figure 1 shows the hierarchy schema and the elements of the Instrument dimension together with the rollup relations between elements. Dimension $\mathcal{D}$ in Figure 1(b) is inconsistent with respect to the strictness constraint Instrument $\rightarrow$ Classification, that requires that every instrument rollups to a unique classification. In fact, element SX-12 rollups to WIND, following edges $(\mathsf{SX\text{-}12}, \mathsf{AER})$ (Aerophone) and $(\mathsf{AER}, \mathsf{WIND})$, and to PER (Percussion) following the edges $(\mathsf{SX\text{-}12}, \mathsf{SAXO})$ and $(\mathsf{SAXO}, \mathsf{PER})$. □



(a) Hierarchy schema (b) Elements and rollup relations

Fig. 1. Instrument DW

When a dimension fails to satisfy its strictness constraints it is possible to compute its minimal repairs [2]. In this paper we present a system that implements logic programs to obtain minimal repairs.

II. The Multidimensional Model

A *hierarchy schema* $\mathcal{H}$ consists of a pair $(\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$, where $(\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$ is an acyclic directed graph. Vertices in the set $\mathcal{C}_\mathcal{H}$ are categories and the edges $\nearrow_\mathcal{H}$ represent the child/parent relations between categories. The transitive and reflexive closure of $\nearrow_\mathcal{H}$ is denoted by $\nearrow^*_\mathcal{H}$. The set of categories $\mathcal{C}_\mathcal{H}$ contains a distinguished *top* category called $\mathsf{All}_\mathcal{H}$, which is reachable from every other category in $\mathcal{C}_\mathcal{H}$ and has no outgoing edges, that is, there is no category $\mathsf{c}_i \in \mathcal{C}_\mathcal{H}$ such that $(\mathsf{All}_\mathcal{H}, \mathsf{c}_i) \in \nearrow_\mathcal{H}$ and for every $\mathsf{c}_j \in \mathcal{C}$, $(\mathsf{c}_j, \mathsf{All}_\mathcal{H}) \in \nearrow^*_\mathcal{H}$. Sometimes, we will write $c_a \nearrow_\mathcal{H} c_b$ instead of $(c_a, c_b) \in \nearrow_\mathcal{H}$.

Example 2: The hierarchy schema $\mathcal{H} = (\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$, depicted in Figure 1(a), is as follows:

- $\mathcal{C}_\mathcal{H}$ = {Instrument, Family, Type, Classification, All};
- $\mathsf{All}_\mathcal{H}$ = All; and
- $\nearrow_\mathcal{H}$={ (Instrument, Family), (Instrument, Type), (Type, Classification), (Family, Classification), (Family, All), (Type, All)}. □

A *dimension* $\mathcal{D}$ is a tuple $(\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Cat}_\mathcal{D}, <_\mathcal{D})$, where: (i) $\mathcal{H}_\mathcal{D} = (\mathcal{C}_{\mathcal{H}_\mathcal{D}}, \nearrow_{\mathcal{H}_\mathcal{D}})$ is a hierarchy schema. (ii) $\mathcal{E}_\mathcal{D}$ is a set of constants, called elements. (iii) $\mathsf{Cat}_\mathcal{D} : \mathcal{E}_\mathcal{D} \rightarrow \mathcal{C}_{\mathcal{H}_\mathcal{D}}$ is a function that defines to which category each element in $\mathcal{E}_\mathcal{D}$ belongs. (iv) the relation $<_\mathcal{D} \subseteq \mathcal{E}_\mathcal{D} \times \mathcal{E}_\mathcal{D}$ represents the child/parent relations between elements of different categories. We denote by $<^*_\mathcal{D}$ the reflexive and transitive closure of $<_\mathcal{D}$. The following conditions hold: (i) $\mathsf{all}_\mathcal{D}$ is the only element in category $\mathsf{All}_{\mathcal{H}_\mathcal{D}}$ (ii) for all pair of elements $a, b \in \mathcal{E}_\mathcal{D}$ if $a <_\mathcal{D} b$ then $\mathsf{Cat}_\mathcal{D}(a) \nearrow_{\mathcal{H}_\mathcal{D}} \mathsf{Cat}_\mathcal{D}(b)$. Condition (ii) ensures that the child/parent relation ($<_\mathcal{D}$) only connects elements of categories that are connected in the schema.

Example 3: Let $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Cat}_\mathcal{D}, <_\mathcal{D})$ be the dimension given in Figure 1(b). Then $\mathcal{D}$ is defined as follows:

- $\mathcal{E}_\mathcal{D} = \{$all, SX-12, YAMAHA-S, SYNTH, SAXO, AER, ELEC, WIND, PER$\}$;
- $\mathsf{all}_\mathcal{D} =$ all;
- $\mathsf{Cat}_\mathcal{D} = \{$all $\mapsto$ All, SX-12 $\mapsto$ Instrument, YAMAHA-S $\mapsto$ Instrument, SYNTH $\mapsto$ Type, SAXO $\mapsto$ Type, AER $\mapsto$ Family, ELEC $\mapsto$ Family, WIND $\mapsto$ Classification, PER $\mapsto$ Classification$\}$; and
- $<_\mathcal{D} = \{$(SX-12,AER), (YAMAHA-S,ELEC), (SX-12,SAXO), (YAMAHA-S,SYNTH), (AER,WIND), (ELEC,PER), (SAXO, PER), (SYNTH,PER), (WIND,all), (PER,all)$\}$. □

There is a rollup relation denoted by $\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ for each pair of categories $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_{\mathcal{H}_\mathcal{D}}$ such that $\mathsf{c}_i \nearrow^*_{\mathcal{H}_\mathcal{D}} \mathsf{c}_j$. This relation is defined as follows:

$$\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j) = \{(a, b) | \mathsf{Cat}_\mathcal{D}(a) = \mathsf{c}_i, \mathsf{Cat}_\mathcal{D}(b) = \mathsf{c}_j \text{ and } a <^*_\mathcal{D} b\}$$

Example 4: Consider the dimension in Figure 1(b), the following are rollup relations:

- $\mathcal{R}_\mathcal{D}(\mathsf{Instrument}, \mathsf{Family}) = \{$(SX-12,AER),(YAMAHA-S, ELEC)$\}$
- $\mathcal{R}_\mathcal{D}(\mathsf{Family}, \mathsf{Classification}) = \{$(AER, WIND), (ELEC, PER)$\}$. □

The rollup relations can be classified into strict and covering. The rollup relation $\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ is said to be *strict* if it is a function, this is, for all elements x, y, z in $\mathcal{E}$, if $(x, y) \in \mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ and $(x, z) \in \mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ then $y = z$. The rollup relation $\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ is said to be *covering* if for all elements $e \in \mathcal{E}$ such that $\mathsf{Cat}(e) = \mathsf{c}_i$, there exists an element $e' \in \mathcal{E}$ such that $(e, e') \in \mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$. A dimension is *strict* if all its rollup relations are strict. Otherwise, the dimension is said to be *non-strict*. Similarly, we use the notions of *covering* and *non-covering* dimensions.

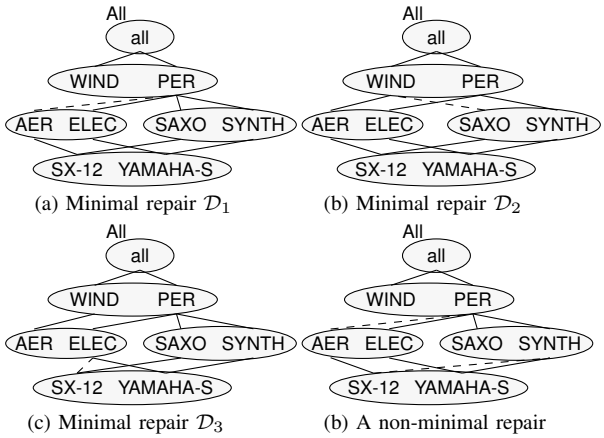
For strict and covering dimensions [6] summarizations are always correct, that is, operations that aggregate facts between two categories that are connected in the hierarchy schema. However, in some real situations dimensions fail to satisfy these conditions [9], [12]. In these cases, to derive correct summarizations it is necessary to specify integrity constraints to identify rollup relations that are strict or covering [9].

Let $\mathcal{H} = (\mathcal{C}_\mathcal{H}, \nearrow_\mathcal{H})$ be a hierarchy schema and let $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Cat}_\mathcal{D}, <_\mathcal{D})$ be a dimension such that $\mathcal{H}_\mathcal{D} = \mathcal{H}$.

- A *strictness constraint* over $\mathcal{H}$ is an expression of the form $\mathsf{c}_i \rightarrow \mathsf{c}_j$ where $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_\mathcal{H}$ and $\mathsf{c}_i \nearrow^*_\mathcal{H} \mathsf{c}_j$. The dimension $\mathcal{D}$ *satisfies* the strictness constraint $\mathsf{c}_i \rightarrow \mathsf{c}_j$ if and only if the rollup relation $\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ is strict.
- A *covering constraint* over $\mathcal{H}$ is an expression of the form $\mathsf{c}_i \Rightarrow \mathsf{c}_j$ where $\mathsf{c}_i, \mathsf{c}_j \in \mathcal{C}_\mathcal{H}$ and $\mathsf{c}_i \nearrow^*_\mathcal{H} \mathsf{c}_j$. The dimension $\mathcal{D}$ *satisfies* the covering constraint $\mathsf{c}_i \Rightarrow \mathsf{c}_j$ if and only if the rollup relation $\mathcal{R}_\mathcal{D}(\mathsf{c}_i, \mathsf{c}_j)$ is covering.

A dimension $\mathcal{D}$ satisfies a set of constraints Σ if $\mathcal{D}$ satisfies every constraint in Σ. Otherwise, the dimension $\mathcal{D}$ is inconsistent with respect to Σ.

Example 5: As shown in Example 1, the dimension $\mathcal{D}$ in Figure 1(b) is inconsistent with respect to the strictness constraint Instrument $\rightarrow$ Classification, that requires that every instrument rollups to a unique classification. This is because element SX-12 reaches more than one element in Classification. However, $\mathcal{D}$ satisfies the following set Σ of strictness constraints: $\{$Instrument $\rightarrow$ Family, Instrument $\rightarrow$ Type, Family $\rightarrow$ Classification, Type $\rightarrow$ Classification$\}$. □



(a) Minimal repair $\mathcal{D}_1$ (b) Minimal repair $\mathcal{D}_2$

(c) Minimal repair $\mathcal{D}_3$ (b) A non-minimal repair

Fig. 2. Repairs for Dimension in Figure 1

III. Repairing Dimensions in Data Warehouses

Intuitively, a *minimal* repair of a dimension $\mathcal{D}$ is a new dimension $\mathcal{D}'$ over the same schema of $\mathcal{D}$, that satisfies the strictness and covering constraints and that is obtained by applying a minimal number of *repair operations* over $\mathcal{D}$. The repair operations correspond to deletions and insertions of edges between elements [2], [1].

Example 6: Figure 2 shows the minimal repairs for dimension $\mathcal{D}$ in Figure 1(b).

- $\mathcal{D}_1$ is obtained from $\mathcal{D}$ by deleting edge (AER,WIND) and inserting (AER,PER).
- $\mathcal{D}_2$ is obtained by deleting edge (SAXO,PER) and inserting (SAXO,WIND).
- $\mathcal{D}_3$ is obtained by deleting edge (SX-12,AER) and inserting (SX-12,ELEC).

All the repairs are minimal since they are obtained by performing a minimal number of repair operations (one insertion and one deletion). Dimension $\mathcal{D}_4$ is not minimal since it is obtained by applying four repair operations. □

To formally define repairs, we need to introduce the following concept. Given two dimensions $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathsf{Cat}_\mathcal{D}, <_\mathcal{D})$

and $\mathcal{D}' = (\mathcal{H}_{\mathcal{D}'}, \mathcal{E}_{\mathcal{D}'}, \mathsf{Cat}_{\mathcal{D}'}, <_{\mathcal{D}'})$, the distance between them, $dist(\mathcal{D}, \mathcal{D}')$, is defined as $|(<_{\mathcal{D}'} \setminus <_{\mathcal{D}}) \cup (<_{\mathcal{D}} \setminus <_{\mathcal{D}'})|$. The distance $dist(\mathcal{D}, \mathcal{D}')$ is the size of the symmetric difference between the child/parent relations of the two dimensions.

Definition 1: [1] Let $\mathcal{D} = (\mathcal{H}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}}, \mathsf{Cat}_{\mathcal{D}}, <_{\mathcal{D}})$ be a dimension and Σ be a set of integrity constraints over $\mathcal{H}_{\mathcal{D}}$.

- A *repair* of $\mathcal{D}$ with respect to Σ is a dimension $\mathcal{D}' = (\mathcal{H}_{\mathcal{D}'}, \mathcal{E}_{\mathcal{D}'}, \mathsf{Cat}_{\mathcal{D}'}, <_{\mathcal{D}'})$ such that $\mathcal{H}_{\mathcal{D}'} = \mathcal{H}_{\mathcal{D}}$, $\mathcal{E}_{\mathcal{D}'} = \mathcal{E}_{\mathcal{D}}$, $\mathsf{Cat}_{\mathcal{D}'} = \mathsf{Cat}_{\mathcal{D}}$, and $\mathcal{D}'$ satisfies Σ.
- A *minimal* repair of $\mathcal{D}$ with respect to Σ is a repair $\mathcal{D}'$, such that $dist(\mathcal{D}, \mathcal{D}')$ is minimal among all the repairs of $\mathcal{D}$ with respect to Σ. □

In this definition, the set of elements in each category of the original dimension is preserved over all the repairs, that is deletions or additions of new elements are not allowed.

Computing minimal repairs of inconsistent dimensions is an NP-hard problem [2]. However, we can specify and compute them by using Datalog programs with negation and weak constraints under the stable model semantics [11]. Repair programs to compute minimal repairs of dimensions with respect to strictness and covering constraints were formally introduced in [2], and presented in [1]. It has been shown that there is a one-to-one correspondence between the *best models* of repair programs and the minimal repairs of inconsistent dimensions [2]. The *best models* of repair programs are stable models that minimize the number of violations of weak constraints. We will illustrate repairs programs in Section IV through an example.

IV. Architecture of the System

DWF is a web-based system developed in HTML[1] and PHP[2] version 5.2.8. It interacts with a DW represented by special relational tables [2] and stored in MySQL DBMS [3] version 5.1.3. *DWF* works under both Linux and Windows operative systems. For an optimal performance, the system requires 1 GB of RAM, a two core processor, and a disc with 5400 rpm.

Dimensions in *DWF* are supposed to be consistent with respect to their covering constraints but may fail to satisfy their strictness constraints. *DWF* generates automatically all the strictness constraints for a dimension, checks consistency of them and fixes inconsistencies with respect to them. To do this, it implements the repair programs presented in [2], [1]. *DWF* interacts with DLV system[4] to compute the best models of the repair programs [11].

The system is composed by the following six modules:

1) *DW's connector*, which allows to connect to the DW and load the DW to the system, this is, it loads hierarchy schemas and strictness constraints.
2) *DW's manager*, that permits to add/delete/update dimensions.
3) *Strictness constraint generator*, that generates the strictness constraints for a dimension.
4) *Consistency checker*, that verifies if a dimension is consistent with respect to its strictness constraints.
5) *Repair generator*, that generates repair programs to compute minimal repairs of dimensions and also computes the *best models* of programs.
6) *DW's Fixer* that restores consistency of dimensions by performing the minimal changes selected by the user.

The main module of the system is the *Repair generator* module, which constructs the repair programs to compute minimal repairs. As an illustration, consider the dimension $\mathcal{D}$ in Figure 1(b) that is inconsistent with respect to the strictness constraint Instrument → Classification. *DWF* will generate a repair program that contains the following rules:

1) Instrument(SX-12). Instrument(YAMAHA-S). Family(AER). Family(ELEC). Type(SAXO). Type(SYNTH). All(all). Classification(WIND). Classification(PER).
2) R(SX-12, AER, Instrument, Family).
   R(YAMAHA-S, ELEC, Instrument, Family).
   R(SX-12, SAXO, Instrument, Type).
   R(YAMAHA-S, SYNTH, Instrument, Type).
   R(AER, WIND, Family, Classification).
   R(ELEC, PER, Family, Classification).
   R(SAXO, PER, Type, Classification).
   R(SYNTH, PER, Type, Classification).
   R(WIN, all, Classification, All).
   R(PER, all, Classification, All).
3) $R'(X, Y,$ Instrument, Family$) \leftarrow$ Instument(X), Family(Y), choice$((X,$ Family$)(Y))$.
   $R'(X, Y,$ Instrument, Type$) \leftarrow$ Instument(X), Type(Y), choice$((X,$ Type$)(Y))$.
   $R'(X, Y,$ Family, Classification$) \leftarrow$ Family(X), Classification(Y), choice$((X,$ Classification$)(Y))$.
   $R'(X, Y,$ Type, Classification$) \leftarrow$ Type(X), Classification(Y), choice$((X,$ Classification$)(Y))$.
   $R'(X, Y,$ Classification, All$) \leftarrow$ Classification(X), All(Y), choice$((X,$ All$)(Y))$.
4) $RT'(X, Y, N_1, N_2) \leftarrow R'(X, Y, N_1, N_2)$.
   $RT'(X, Z, N_1, N_3) \leftarrow RT'(X, Y, N_1, N_2), R'(Y, Z, N_2, N_3)$.
5) $\leftarrow RT'(X, Y, N_1, N_2), RT'(X, Z, N_1, N_2), Y \neq Z$.
6) $Ins(X, Y, N_1, N_2) \leftarrow R'(X, Y, N_1, N_2), not\ R(X, Y, N_1, N_2)$.
   $Del(X, Y, N_1, N_2) \leftarrow R(X, Y, N_1, N_2), not\ R'(X, Y, N_1, N_2)$.
7) $\Leftarrow Ins(X, Y, N_1, N_2)[1:1]$.
   $\Leftarrow Del(X, Y, N_1, N_2)[1:1]$.

Rules in (1) capture the elements in categories of $\mathcal{D}$. Rules in (2) are the rollup relations in $\mathcal{D}$. The rest of the rules in the repair program are needed to compute the dimension repairs. The program generates all possible dimensions $\mathcal{D}' = (\mathcal{H}_{\mathcal{D}'}, \mathcal{E}_{\mathcal{D}'}, \mathsf{Cat}_{\mathcal{D}'}, <_{\mathcal{D}'})$ such that $\mathcal{H}_{\mathcal{D}'} = \mathcal{H}_{\mathcal{D}}$, $\mathcal{E}_{\mathcal{D}'} = \mathcal{E}_{\mathcal{D}}$, and $\mathsf{Cat}_{\mathcal{D}'} = \mathsf{Cat}_{\mathcal{D}}$ (see Definition 1). Rules in (3) ensure that every element in the dimensions has a unique parent in every parent category. This parent is obtained by using the choice operator [5], where choice$((X, c_j), (Y))$ will assign a unique

[1] http://www.w3.org/MarkUp/
[2] http://www.php.net/
[3] http://www.mysql.com/
[4] http://www.dbai.tuwien.ac.at/proj/dlv/

value to Y in each stable model for each combination (X, c_j). Then, rules in (4) and (5) discard the dimensions that are not strict. Finally, using rules in (6) and (7) the program chooses the dimensions that are minimal repairs. Rules in (7) are called weak constraints, and allow us to keep track of the number of insertions/deletions needed to restore consistency [1].

Then, *DWF* will compute the best models of the repair program, i.e., the stable models that minimize the number of violations of weak constraints. The repair program for the inconsistent dimension $\mathcal{D}$ in Figure 1(b) has the following three best models[5]:

- $\mathcal{M}_1$= $\{Del(\text{AER, WIND, Family, Classification}), Ins(\text{AER, PER, Family, Classification})\}$.
- $\mathcal{M}_2$= $\{Del(\text{SAXO, PER, Type, Classification}), Ins(\text{SAXO, WIND, Type, Classification})\}$,
- $\mathcal{M}_3$= $\{Del(\text{SX-12, AER, Instrument, Family}), Ins(\text{SX-12, ELEC, Instrument, Family})\}$,

The total weight of each best model is 2 and corresponds to the distance between each of the repairs and dimension $\mathcal{D}$. The insertions/deletions operations specified by the models generate, respectively, the minimal repairs shown in Figure 2.

Finally, the system retrieves the three options to restore consistency and then, via the module *DW's Fixer*, the user can select the most suitable option. The latter module will materialize the changes described in the selected best model.

V. Interface

Figure 3 shows the main screen of the *DWF*system.

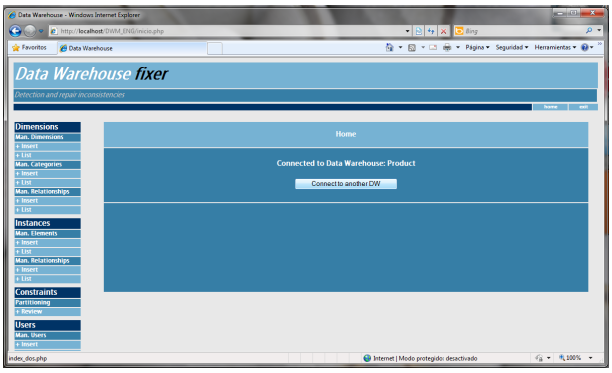


Fig. 3. *DWF* Main screen

Figure 4 shows information about dimension in Figure 1 which is inconsistent with respect to the strictness constraint Instrument $\rightarrow$ Classification. It also shows the elements of Instrument involved in the inconsistencies.

Figure 5 shows the repair options. As expected, there are three alternative repair operations to restore consistency of the dimension. The user can materialize, and in this way, fix the dimension by clicking one, and only one, alternative.

[5]We show only the Ins and Del atoms which show the modifications needed to restore consistency.

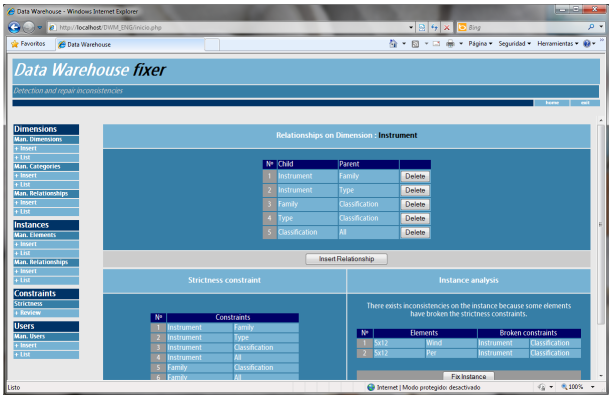


Fig. 4. *DWF*: Checking consistency

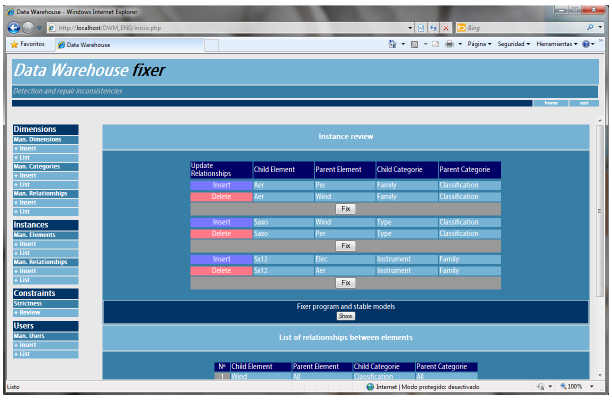


Fig. 5. *DWF*: Repairing an inconsistent dimension

VI. Conclusions

DWF computes dimension repairs with respect to strictness constraints by evaluating logic programs with stable model semantics. As far as we know, *DWF* is the first system that computes repairs of DW's dimensions. We left as a future work the experimentation of the system, and also the analysis of optimizations, since it is known that the evaluation of logic programs over large data sets may be inefficient [3].

Acknowledgments

This project was funded by FONDECYT grant #11070186. Currently, Mónica Caniupán is funded by University of Bío-Bío (Grant DIUBB 110115 2/R).

References

[1] Bravo, L., and Caniupán, M. and Hurtado, C. Logic Programs for Repairing Inconsistent Dimensions in Data Warehouses. In *AMW'10*, 2010.

[2] Caniupán, M., and Bravo, L. and Hurtado, C. Repairing Inconsistent Dimensions in Data Warehouses Submitted to Data Knowl. Eng., June 2010.

[3] Caniupán, M., and Bertossi, L. The Consistency Extractor System: Answer Set Programs for Consistent Query Answering in Databases *Data Knowl. Eng.*, 6(69): 545–572, 2010.

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

[5] Giannotti, F., and Greco, S., and Saccà, D., and Zaniolo, C. Programming with Non-determinism in Deductive Databases. *AMAI*, 19(1-2):97–125, 1997.

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

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

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

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

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

[11] Leone, N., and Pfeifer, G., and Faber, W., and Eiter, T., and Gottlob, G., and Koch, Ch., and Mateis, C., and Perri, S. and Scarcello, F. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

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

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