A Multidimensional Data Model with Subcategories for Flexibly Capturing Summarizability

Sina Ariyan
Carleton University
School of Computer Science
Ottawa, Canada
mariyan@scs.carleton.ca

Leopoldo Bertossi
Carleton University
School of Computer Science
Ottawa, Canada
bertossi@scs.carleton.ca

ABSTRACT

In multidimensional (MD) databases and data warehouses we commonly prefer instances that have summarizable dimensions. This is because they have good properties for query answering. Most typically, with summarizable dimensions, precomputed and materialized aggregate query results at lower levels of the dimension hierarchy can be used to correctly compute results at higher levels of the same hierarchy, improving efficiency. Being summarizability such a desirable property, we argue that some established MD models cannot properly model the summarizability condition, and this is a consequence of the limited expressive power of the modeling languages. We propose an extension to the Hurtado-Meldelzon (HM) MD model with subcategories, the EHM model, and show that it allows to capture the summarizability. We propose an efficient algorithm that, for a given cube view (i.e. MD aggregate query) in an EHM database, determines from which minimal subset of precomputed cube views it can be correctly computed. Finally, we show how the EHM can be implemented with minor modifications to the familiar ROLAP schemas.

Keywords

multidimensional databases and models, data warehouses, summarizability, query answering, categories, ROLAP

1. INTRODUCTION

Data warehouses form a particular class of multidimensional databases (MDDBs), and combine data from different sources, for analysis and decision support. MDDBs represent data in terms of *dimensions* and *fact tables*. A dimension is represented in terms of a *dimension schema*, i.e. a hierarchy (more generally, a lattice) of category names, plus a *dimension instance* that assigns extensions to the category names, organizing them in a hierarchy that parallels the category hierarchy. A row in the fact table corresponds to a measurement that is taken at the intersection of the categories at the bottom of the dimensions: a quantity that is given context through base dimension values. This MD organization allows users to aggregate data at different levels of granularity, corresponding to different levels of categories in the hierarchies. Aggregation is the most important operation in OLAP applications. OLAP queries are complex, may involve massive data, and also aggregation. Therefore, fast query answering is a crucial task in OLAP [24].

SSDBM '13, July 29 - 31 2013, Baltimore, MD, USA

Some model of MDDBs have been proposed in the literature [8, 23], but we adopt as basis for our research the Hurtado-Mendelzon (HM) model of MDDBs [15, 19]. It is an established and extensively investigated model, and it provides solid and clear foundations for further investigation of MDDBs.

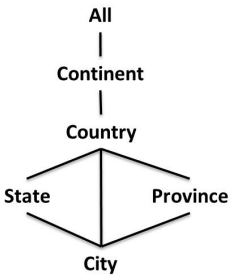
Example 1. We represent data about a company sales for different *times* and *locations* in Figure 1. We illustrate there only the `Location` dimension, represented as a hierarchy of places where the company has branches. The `Location` dimension schema is in Figure 1a, and a possible instance for it is shown in Figure 1b.

In this schema, `City`, `State`, etc., are *categories*. `Country` is a *parent* of categories `State` and `Province`; and an *ancestor* of category `City`. Category `City` is a *base category*, i.e. it has no children. At the instance level, element `LA` of `City` *rolls up* to `USA`, as an ancestor element in category `Country`. `LA` and `Vancouver` are *base elements*. As we can see, both the schema and the instance are endowed each with compatible partial orders (lattices) that are directed from the bottom to the top.

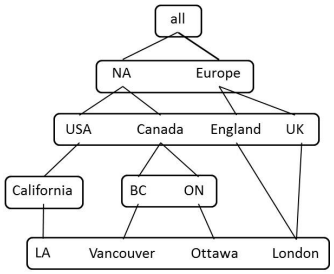
Figure 1c shows the fact table containing data about sales for cities (of the base category) at different times. □

Due to the possibly very large volumes of data in MDDBs, computation from scratch of aggregate queries should be avoided whenever possible. Ideally, aggregate query results at lower levels should be used to correctly calculate results at higher levels of the hierarchy. If this is the case, we say -informally for the moment (precise definition will follow)- that the dimension instance is *summarizable*. This notion can be applied to an instance in local terms: if a parent category and a set of its descendant categories allow for this, we say that the parent is *(locally) summarizable* from those descendants. For this to happen, every base element that rolls up to some element in the parent category must pass through one and only one element from the descendant categories [19]. As an example, in the instance in Figure 1b, category `All` is summarizable from `Continent`, but `Country` is not summarizable from `Province`: the base elements `LA` and `London` do not pass through any element from `Province` when rolling up to `Country`. In the absence of summarizability, a multidimensional database will either return incorrect query results when using pre-computed views, or lose efficiency by having to compute the answers from scratch [20, 27].

We can see that summarizability may take different shapes: local and global, and in the latter case relative to *all* the descendants of a category or only to a subclass of them. Accordingly, we say that a dimension is *strict-summarizable* if every category is (locally) summarizable from each one of its descendant categories. This is the



(a) `Location` schema



(b) `Location` instance

City	Date	Sales
Ottawa	Jan 1,12	6000
Vancouver	Jan 1,12	4500
London	Jan 1,12	10000
Ottawa	Jan 2,12	5500
London	Jan 2,12	1400
LA	Jan 2,12	3000

(c) Fact table

Figure 1: Sales database

concept that has been the focus of previous research in the literature [6, 7, 11, 10, 19, 29]. In particular, it has been established that for a dimension to be strict-summarizable, it must satisfy three conditions [20, 29]. First, the dimension schema must have a single *base category*, which contains the all elements that have no children. Second, the dimension instance must be *strict*, i.e. each element in a category must have at most one parent in each upper category. Third, the instance has to be *homogeneous* (aka. *covering*), i.e. each element in a category has at least one parent element in each parent category. The last two conditions can be seen as semantic constraints on a dimension instance. We refer to the combination of these three conditions as the *strict-summarizability conditions*. The `Location` dimension in Figure 1 has a single base category, `City`. But the instance is not strict: `London` rolls up to the two different elements in `Country`: `England` and `UK`. It is also non-homogeneous, i.e. heterogeneous: `London` does not roll up to any element in `State` or `Province`.

Dimensions that do not satisfy strictness or homogeneity (or both) are said to be *inconsistent*. The `Location` dimension instance in Figure 1 is inconsistent. It violates both strictness and homogeneity. Dimensions may become inconsistent for several reasons, in particular, as the result of a poor design or a series of update operations [22].

We argue that the strict-summarizability conditions are too restrictive, and imposing them will filter out a wide range of useful models and instances. This includes any dimension that has multiple base categories. However, there are other cases. As another example, the instance in Figure 2 shows that category `Country` is not summarizable from `State`, nor from `Province`. Therefore, the dimension is not strict-summarizable. However, `Country` is summarizable from the combination of `State` and `Province`. Therefore, we can still take advantage of pre-computed cube views at the level of `State` and `Province` to compute aggregate results for `Country`. We will redefine the notion of summarizability for a dimension instance to take multi-child aggregation into consideration.

The restrictions imposed by strict-summarizability are not the whole problem. It is not only that an instance at hand fails to be summarizable. Actually, there are many commonly used dimensions for which the HM metamodel (i.e. the framework that supports

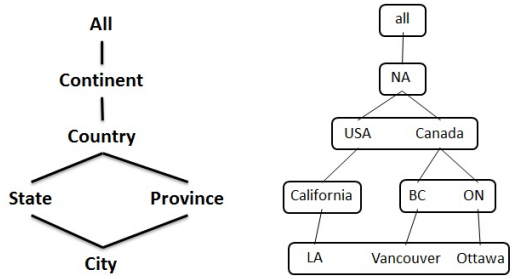


Figure 2: A dimension that will be filtered out by the imposition of strict-summarizability conditions

the creation of specific HM models) fails to accommodate an MD model, including instances, that satisfies summarizability. This is because any HM model that captures the external reality will be bound, already by the sheer MD schema used, to have instances that do not satisfy the summarizability conditions expressed above. An example is given by Figure 2: when capturing in an HM model a real-world assumption, such as having `State` and `Province`, at the same level, as immediate *and partitioning* ancestors of `City`, there is no HM model that can capture this domain in a summarizable way. Every reasonable instance (e.g. with non-empty categories) will not be summarizable.

One way to approach the problem of modeling such domains consists in making changes to the dimension schema and/or the dimension instance to restore strictness and homogeneity. In accordance with the area of *consistent query answering* (CQA) in relational databases [1, 4, 5], a resulting dimension instance could be called a *repair* of the original one. A *minimal repair* (we usually and simply call a "repair") is one that minimally differs (in some prescribed sense of minimality) from the original dimension and satisfies strictness and homogeneity [2, 6]. In some cases, this repair process might lead to unnatural repairs, that do not quite conform to the external reality.

Instance-based repairs have been introduced and studied in [6, 7, 11, 9, 29], and in some sense, but not as repairs, in [28, 18]. In this case, consistency is restored by modifying data elements or links between them in the original dimension instance. For this reason, these repairs are also called *data repairs*. These repairs have been proposed to deal mainly with non-strictness, and usually assuming homogeneity [6, 9]. When both non-strictness and heterogeneity are addressed [7, 11, 29], the latter has been confronted through the insertion of null values [18, 28, 29].

Schema-based repairs (aka. *structural repairs*) have been formally proposed and studied in [2], and under different forms and names, in [18, 19, 21]. They modify the schema of an inconsistent dimension instance by adding or removing categories and/or links between them, to restore strictness and homogeneity. These changes on the schema affect an instance in that elements of categories have to be redistributed in the categories of the modified schema (for the links between elements to parallel the links between categories). However, under schema-based repairs, there are no "data changes" (as with data repairs), in the sense that the data elements and also the edges connecting them persist.

Data repairs have been the main focus of previous work on repairing MDDBs. However, there exist real world domains, such as the one in Example 1, where this approach may produce an unnatural solution, e.g. for the instance in Figure 1b), one that connects `London` to `California` in `State`, and to `BC` or a null in `Province`. Of course, additional restrictions could be imposed on the repair

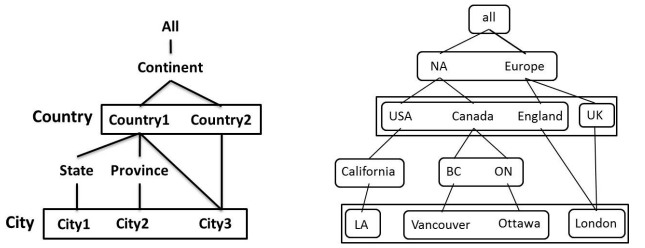


Figure 3: An EHM `Location` dimension

process, like not allowing such links, but in some cases there might be no repair then.

With data repairs, a single change on a dimension instance directly results in several changes on records in the underlying database (the number of changes depends on the underlying ROLAP schema, and is usually larger when changes are made at higher levels of the instance [29]). With structural repairs this is not the case, which is advantageous for users who do not want their records changed. On the other hand, with structural repairs, some sort of query reformulation becomes necessary since old queries may refer to categories of the old schema that may have been changed or disappeared (with consequent changes on the relational schemas).

In this work, as an alternative to repairs for confronting the modeling problem mentioned above, we introduce and investigate an extension of the HM metamodel. This extension allows us to produce *extended HM (EHM) models*. The main ingredient of this extension is the addition of *subcategories*. Figure 3 shows an EHM model of the `Location` dimension. Subcategories are shown as rectangles inside a category (when a subcategory coincides with its containing category, we omit the inner rectangle). The idea, that we develop in this work, is that, by introducing subcategories `Country1` and `Country2`, category `Continent` now becomes summarizable from its direct lower level, in this case from subcategory `Country1`. We did not have summarizability in any HM model of this dimension.

We investigate the extended model as a way to directly model MDDBs, with some clear advantages over HM models. Most importantly, EHM is -in a precise technical sense- more expressive than HM for modeling MDDBs subject to summarizability conditions. In particular, it follows that a summarizable HM model is still a summarizable EHM model, without any changes. Also, there are common real-world examples for which no summarizable HM model exists, but can be modeled using a summarizable EHM model.

We establish that EHM models have a positive computational feature: Given any category C and an arbitrary subset, K, of its descendant categories in an EHM dimension, one can decide whether C is summarizable from K without processing the instance (which is usually very large). Furthermore, this decision algorithm can be extended in order to compute for a category C, the class $\mathcal{K}$ of all sets K of descendants from which it is summarizable. EHM models enjoy this feature because subcategories reflect the exact structure of the dimension instance (but at a much higher level or granularity, which reduces size). As expected, this behavior comes at the expense of imposing additional semantic constraints on dimension instances. This will have an impact on the population and maintenance of the dimension instance. However, this cost is more than offset by the benefits for OLAP query answering. More precisely, when confronted with an aggregate query $\mathcal{Q}$, it will be possible to determine (at a preprocessing step) the most appropriate set of pre-computed aggregate views at lower levels that can be used to efficiently answer $\mathcal{Q}$.

Notice that, for an arbitrary, non-extended HM model, a decision procedure and an algorithm for computing "summarizable classes of children" as for EHM models will also depend on the dimension instance, which is usually very large. Therefore, a similar preprocessing step becomes impractical for aggregate query answering in HM models. It should be noted that although the EHM metamodel is an extension of the HM metamodel, not necessarily an HM dimension (including the schema and instance) is an EHM dimension, because the latter are subject to additional conditions, as mentioned above. As a consequence, the algorithms for the EHM model just described do not necessarily apply to HM models, as expected.

The conditions on EHM dimension instances have to be imposed and maintained. We provide an algorithm for populating a EHM dimension instance. It is based on first specifying the categories, and then adding data links one by one. In some cases, that seem to be rare in practice, this may require updating the subcategory schema. More precisely, splitting a subcategory in two (within the same category), and creating a new link to the upper subcategories. Actually, this approach opens the possibility of creating EHM models from scratch.

We show that the extended HM model does not produce complications with the implementation of MDDBs as/on relational databases (ROLAP) [25]. Actually, our extended HM models can be captured by the well-known and familiar ROLAP schemas, like star or snowflake, with minor modifications.

As stated earlier, EHM models have better summarizability properties that HM models. In particular, we can take advantage of more pre-computed views (from lower levels) when answering higher level queries, with a significant impact on the efficiency of query answering. We will provide experimental support for this claim.

Summarizing, in this paper we make the following contributions:

1. We introduce and formalize the Extended HM models (EHM models) for MDDBs (Section 2). We also specify when an HM model can be considered as particular kind of EHM model.

2. We introduce a new, less restrictive notion of summarizability that applies to both HM and EHM dimensions (Section 3). We show that: (a) Summarizable HM models are still summarizable as EHM models. (b) There are sensible real-world examples for which no summarizable HM model exists, but that can be modeled using a summarizable EHM model, showing a difference in expressive power.

3. We present the algorithms for EHM dimensions previously announced, in particular, to determine from which precomputed cube views a cube view (i.e. aggregate query) can be correctly computed, for summarizability reasons (Section 4.2).

4. We show how EHM dimension instances can be populated from scratch, assuming categories (but not subcategories) are already specified (Section 5.1).

5. We show that EHM models can be implemented with the usual ROLAP schemas, with minor modifications (Section 5.2).

6. We provide a formal characterization of expressiveness for classes of dimension models. We establish that the class of summarizable HM dimensions is less expressive than that of summarizable EHM dimensions (Section 6).

7. In Section 7 we show our experiments.

An extended version of this paper can be found at [3]. It contains an Appendix with the proofs missing here and our instance population algorithm.

2. THE EXTENDED HM MODEL

An EHM *dimension schema* $\mathfrak{S}$ is a tuple of the form $\langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$, where U is a possibly infinite set (the underlying data domain), $\mathcal{C}$ is a finite set of categories (or better, category names), $\mathcal{S}$ is a finite set of *subcategories* (or names therefor), $\langle \mathcal{S}, \nearrow \rangle$ is a directed acyclic graph, and $\theta : \mathcal{S} \to \mathcal{C}$ is a total and onto (surjective) function. $\mathcal{C}$ and $\mathcal{S}$ are disjoint, but when a category has a single subcategory (as determined by θ), we will sometimes identify them. Notice that $\nearrow$ is a binary relation between subcategories. Its transitive and reflexive closure is denoted by $\nearrow^*$. We make the additional assumption that any two subcategories assigned to a same category are $\nearrow^*$-incomparable.

Every dimension schema has a distinguished top category, `All`, with a single subcategory, also denoted by `All`, which is reachable from every other subcategory: For every subcategory $s \in \mathcal{S}$, $s \nearrow^* \texttt{All}$ holds. For a subcategory s, a subcategory s' for which $s' \nearrow s$ ($s' \nearrow^* s$) holds is called a child (resp. descendant) of s. Subcategories that have no descendants are called *base subcategories*. The graph structure on U induces a graph structure on C, which we also denote with $\nearrow$ (and $\nearrow^*$): For $c, c' \in C$, $c \nearrow c'$ holds iff there are s, s' with $\theta(s) = c, \theta(s') = c'$, and $s \nearrow s'$. Accordingly, we can talk about categories that are children or descendants of a category. Finally, we also assume that every schema contains a unique *base category*, i.e. a category with no children. However, this base category may contain one or more *base subcategories*.

Given a dimension schema $\mathfrak{S}$, a *dimension instance* for the schema $\mathfrak{S}$ is a tuple $\mathcal{D} = \langle \mathcal{M}, \delta, < \rangle$, where $\mathcal{M}$ is the finite subset of U, $\delta : M \to \mathcal{S}$ is a total function (assigning data elements to subcategories), and $<$ is a binary relation on $\mathcal{M}$ that parallels relation $\nearrow$ on subcategories: $e_1 < e_2$ iff $\delta(e_1) \nearrow \delta(e_2)$. If $\theta(\delta(e)) = c$, then we also say by extension that $e \in c$. Element $\texttt{all} \in \mathcal{M}$ is the only element of subcategory `All`.

The transitive and reflexive closure of $<$ is denoted with $<^*$, and can be used to define the *roll-up relations* for any pair of categories c_i, c_j: $\mathcal{R}^{c_j}_{c_i}(\mathcal{D}) = \{(e_i, e_j) \mid e_i \in c_i, e_j \in c_j, \text{ and } e_i <^* e_j\}$. Similarly, roll-up relations can be associated to pairs of subcategories or pairs formed by a category and a subcategory (usually of some other category). Notice that in any case where it is not the case that $c_i <^* c_j$, then $\mathcal{R}^{c_j}_{c_i}(\mathcal{D})$ is the empty relation. When $c_i = c_j$, it is the identity relation, with elements of the form (e, e) with $e \in \cup\{\delta(s) \mid s \in \mathcal{S} \text{ and } \theta(s) = c_i\}$.

Definition 1. An instance $\mathcal{D}$ for a schema $\mathfrak{S}$ is *compliant* if, for a pair of subcategories $s_i, s_j \in \mathcal{S}$, $\mathcal{R}^{s_j}_{s_i}(\mathcal{D})$ is a total function. □

The condition above will make sure that the subcategory hierarchy will reflect the structure of the instance. Notice that the requirement on the roll-up relations is not imposed on pairs of categories.

Example 2. The `Location` dimension in Figure 3 can be modeled through the following schema $\mathfrak{S}$:

$$\begin{aligned}
\mathcal{C}_{\texttt{Loc}} &= \{\texttt{City}, \texttt{State}, \texttt{Province}, \texttt{Country}, \texttt{Continent}, \texttt{All}\}.\\
\mathcal{S}_{\texttt{Loc}} &= \{\texttt{City1}, \texttt{City2}, \texttt{City3}, \texttt{State}, \texttt{Province}, \texttt{Country1},\\
&\quad \texttt{Country2}, \texttt{Continent}, \texttt{All}\}.\\
\nearrow &= \{(\texttt{City1}, \texttt{State}), (\texttt{City2}, \texttt{Province}), (\texttt{City3}, \texttt{Country1}),\\
&\quad (\texttt{City3}, \texttt{Country2}), (\texttt{State}, \texttt{Country1}),\\
&\quad (\texttt{Province}, \texttt{Country1}), (\texttt{Country1}, \texttt{Continent}),\\
&\quad (\texttt{Country2}, \texttt{Continent}), (\texttt{Continent}, \texttt{All})\}.
\end{aligned}$$

For example, here, $\theta(\texttt{City1}) = \theta(\texttt{City2}) = \texttt{City} \in \mathcal{C}_{\texttt{Loc}}$. Now, for the corresponding dimension instance $\mathcal{D}$, we have:

$$\begin{aligned}
\mathcal{M}_{\texttt{Loc}} &= \{\texttt{LA}, \texttt{Vancouver}, \texttt{Ottawa}, \texttt{London}, \texttt{California}, \texttt{BC},\\
&\quad \texttt{ON}, \texttt{USA}, \texttt{Canada}, \texttt{England}, \texttt{UK}, \texttt{NA}, \texttt{Europe}, \texttt{all}\}.\\
< &= \{(\texttt{Vancouver}, \texttt{BC}), (\texttt{Ottawa}, \texttt{ON}), (\texttt{LA}, \texttt{California}),\\
&\quad (\texttt{London}, \texttt{England}), (\texttt{London}, \texttt{UK}), (\texttt{BC}, \texttt{Canada}),\\
&\quad (\texttt{ON}, \texttt{Canada}), (\texttt{California}, \texttt{USA}), (\texttt{Canada}, \texttt{NA}),\\
&\quad (\texttt{USA}, \texttt{NA}), (\texttt{England}, \texttt{Europe}), (\texttt{UK}, \texttt{Europe}),\\
&\quad (\texttt{NA}, \texttt{all}), (\texttt{Europe}, \texttt{all})\}.
\end{aligned}$$

Here, for example, $\delta(\texttt{LA}) = \texttt{City1}$. Now, the ancestors in categories `Province` and `Country` of all base elements can be obtained via the following roll-up relations:

$$\begin{aligned}
\mathcal{R}^{\texttt{Province}}_{\texttt{City}}(\mathcal{D}) &= \{(\texttt{Vancouver}, \texttt{BC}), (\texttt{Ottawa}, \texttt{ON})\}.\\
\mathcal{R}^{\texttt{Country}}_{\texttt{City}}(\mathcal{D}) &= \{(\texttt{Ottawa}, \texttt{Canada}), (\texttt{Vancouver}, \texttt{Canada})\\
&\quad (\texttt{LA}, \texttt{USA}), (\texttt{London}, \texttt{England}), (\texttt{London}, \texttt{UK})\}.
\end{aligned}$$

The second one is not a function. However, the subcategory roll-ups are total functions. For example:

$$\mathcal{R}^{\texttt{Country1}}_{\texttt{City2}}(\mathcal{D}) = \{(\texttt{Ottawa}, \texttt{Canada}), (\texttt{Vancouver}, \texttt{Canada})\}.$$

is a total function from (the extension of) subcategory `City2` to subcategory `Country1`. □

Definition 2. An HM dimension schema is an EHM dimension schema $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$, where θ is a bijective function. An HM instance for an HM schema $\mathfrak{S}$ is an instance in the sense of EHM. □

This definition establishes the connection with the (old) HM models [15, 19]. Basically, in an HM dimension, each category has a single subcategory. Notice that an HM instance for its HM schema may not be compliant. However, a compliant HM dimension according to this definition is (isomorphic to) an HM dimension in the sense of [15, 19] that satisfies strictness and homogeneity (as also defined in [15, 19]).

In the rest of this paper, we will be using the terms instance and dimension indistinctly.

3. SUMMARIZABILITY

The most common aggregate queries in DWHs are those that perform grouping by the values of a set of categories from different dimension schemas (known as the *granularity*), and return a single aggregate value per group. These aggregate queries are known as *cube views* [16]. The aggregation is achieved by upward navigation through a path in the dimension schema, which is captured by roll-up relations.

3.1 Classic HM summarizability

In this subsection we review some known concepts and results for summarizability in the HM model. Cube views are defined using distributive aggregate functions [13] with SUM, MAX and MIN being the most common cases. We will use $\Psi_f[c_1, ..., c_n]$ to denote a cube view at granularity $\{c_1, ..., c_n\}$ where f is the distributive aggregate function, and each c_i is a category (name) from a dimension schema $\mathfrak{S}_i$. Most of the time we will concentrate on a single dimension, in which case, the cube view will be of the form $\Psi_f[c]$, with c a category of a dimension schema $\mathfrak{S}$.

Lets assume $\mathcal{D}_1$, ..., $\mathcal{D}_n$ are a set of HM dimensions with schemas $\mathfrak{S}_1$, ..., $\mathfrak{S}_n$ and base categories b_1, ..., b_n, respectively. Cube view $\Psi_f[c_1, ..., c_n]$, where c_k is a category from schema $\mathfrak{S}_k$, can be defined as follows:

$$\Pi_{c_1,...,c_n,f(m)}(\mathcal{R}^{c_1}_{b_1}(\mathcal{D}_1) \bowtie ... \bowtie \mathcal{R}^{c_n}_{b_n}(\mathcal{D}_n) \bowtie F)$$

where m is a measure from fact table F.

A common technique for speeding up OLAP query processing is to pre-compute some cube views and use them for the derivation (or answering) of other cube views. This approach to query answering is correct under the summarizability property, which ensures that higher-level cube views (from the base categories and fact tables) can be correctly computed using cube-views at lower level as if they were fact tables [21]. As shown in Example 3, the reuse of pre-computed cube views has a great impact on query answering performance. Therefore, maximizing summarizability between categories is a crucial optimization task for MDDBs [24].

Example 3. Lets assume the `Sales` database of Figure 1 belongs to a worldwide corporation. The managers of this corporation need to compute its aggregate sales information for every continent between the years 2003 and 2012.

Computing this query directly (i.e. without taking advantage of summarizability) requires a join between three relations: $\mathcal{R}^{\texttt{Continent}}_{\texttt{City}}$, $\mathcal{R}^{\texttt{Year}}_{\texttt{Day}}$ and the fact table. Assuming the fact table contains daily values for 3000 cities over this period of time, these relations will contain more than 3000, 3600 and 10^7 tuples respectively. Obviously this will be a time consuming query.

On the other hand, if we were to compute the same query using a pre-computed cube view, for example $\Psi_{sum}[\texttt{Country}, \texttt{Year}]$, relations that were involved in the new join would be $\mathcal{R}^{\texttt{Continent}}_{\texttt{Country}}$, $\mathcal{R}^{\texttt{Year}}_{\texttt{Month}}$ and $\Psi_{sum}[\texttt{Country}, \texttt{Year}]$ with at most 200, 120 and 24000 tuples respectively. Hence, great improvement in query processing time would be obtained.

This performance improvement would be much more significant in real world examples involving more than two dimensions (e.g. by adding a product dimension) and more detailed base categories (e.g. a `Location` dimension with `Branch` as the base category rather than `City`). □

An HM instance is considered to be summarizable if a cube view on any category c can be computed a from pre-computed cube view *for each one* of its descendant categories c' (as if the cube view for c' were a fact table) [6, 7, 11, 10, 19, 29]. We will refer to this notion of summarizability (i.e. the one used in the literature) as *strict-summarizability*, as we indicated in Section 1. The reason for this is that in Section 3.2 we will relax this notion, providing a less restrictive definition of summarizability.

Definition 3. [29] An HM instance $\mathcal{D}$ is (a) *Strict* iff for all categories c_i,c_j, the roll-up relation $\mathcal{R}^{c_j}_{c_i}(\mathcal{D})$ is a (possibly partial) function. (b) *Homogeneous* iff for all categories c_i,c_j, the roll-up relation $\mathcal{R}^{c_j}_{c_i}(\mathcal{D})$ is *total*. (c) *Consistent* iff it satisfies the two previous conditions; and *inconsistent* otherwise. □

THEOREM 1. [19] An HM instance with a single base category is strict-summarizable iff it is consistent. □

3.2 Flexible summarizability for EHM

The definition of strict-summarizability specified in Section 3.1 can be applied to EHM dimensions as well as HM dimensions. However, in this section, we are going to define a new notion that is more flexible and will cover a wider range of useful models and instances.

In the following, except when otherwise stated, we will consider single EHM dimensions. That is, we consider a dimension schema $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$, and a compliant instance $\mathcal{D} = \langle \mathcal{M}, \delta, < \rangle$. Now we give a definition of summarizability for the extended HM metamodel. However, the new notion of summarizability can be applied to HM dimensions as well as EHM dimensions that may or may not be compliant.

Cube views for EHM dimensions can be defined as in Section 3.1.

Definition 4. (a) Category c is *$\mathcal{K}$-summarizable* with regard to distributive aggregate function f, where $\mathcal{K} = \{c_1, ..., c_n\}$ contains descendants of c, iff every cube view on c can be computed by applying f to pre-computed cube views on $\{c_1, ..., c_n\}$.
(b) $\mathcal{K}$ is a *minimal summarizability set* for c iff c is $\mathcal{K}$-summarizable, and there is no $\mathcal{K}' \subsetneqq \mathcal{K}$ such that c is $\mathcal{K}'$-summarizable.
(c) For any dimension $\mathcal{D}$, the *summarizability sets* of a category c, denoted $SumSets_{\mathcal{D}}(c)$, is a set containing all minimal summarizability sets of c.
(d) Dimension $\mathcal{D}$ is *summarizable* iff every category c in $\mathcal{C}$ is $\mathcal{K}$-summarizable for some subset $\mathcal{K}$ of its child categories. More precisely, for every category c there exists a set $\mathcal{K} \in SumSets_{\mathcal{D}}(c)$, such that $\mathcal{K} \subseteq \{c' | \, c' \nearrow c\}$. □

Notice that in Definition 4, we are not making any reference to subcategories. However, in a compliant EHM dimension, subcategories (and more specifically their roll-ups) will help in the computation of cube views from lower level precomputed cube views. This is demonstrated by the following example.

Example 4. We will demonstrate the effect of subcategories by comparing the HM dimension of Figure 1 with the EHM dimension of Figure 3. Notice that in the HM dimension, a cube view on category `Continent` cannot be computed using a precomputed cube view on `Country` (because element `London` will be counted twice once through `England` and another time through `UK`).

Now, in the case of the EHM dimension of Figure 3, a cube view on category `Continent` can be derived from a precomputed cube view on `Country` by simply joining with the roll-up from subcategory `Country1` as follows:

$$\Psi_f(\texttt{Continent}) = \Psi_f(\texttt{Country}) \bowtie \mathcal{R}^{\texttt{Continent}}_{\texttt{Country1}}. \qquad \square$$

For any dimension, having access to the summarizability sets of categories can be very useful. Suppose, we are given an aggregate query on category c and we want to compute the result using available pre-computed cube views on descendants of c. The class of summarizability sets of c is exactly what we need in order to choose the right set of pre-computed cube views to answer the original query. Therefore, being able to compute and generate these sets can have a great impact on improving query answering performance (as long as the process of generating the summarizability sets does not become more complex than answering the query). Once summarizability sets have been computed, they can be used for different queries until some updates on the dimension instance affect and change the summarizability sets.

Example 5. Consider the `Sales` database of Figure 1 plus a `Date` dimension with categories `Day`, `Month` and `Year`. In this setting, category `Country` is summarizable from the set of categories {`City`, `State`, `Province`} but the latter is not a minimal summarizability set for `Country` because `Country` is also summarizable from {`City`}.

The summarizability sets for categories of the Location dimension are as follows:

$SumSets_{\text{Loc}}(\texttt{All}) = \{\{\texttt{Continent}\}, \{\texttt{Country}\}, \{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Continent}) = \{\{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Country}) = \{\{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{State}) = \{\{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Province}) = \{\{\texttt{City}\}\}$.

Category Continent has only one child, Country, from which it is not summarizable, and therefore, the Location dimension is not summarizable. Actually, we can show that $\Psi_{sum}[\texttt{Continent}, \texttt{Day}]$ cannot be computed by aggregating over the values of cube view $\Psi_{sum}[\texttt{Country}, \texttt{Day}]$. In fact, using the fact table of Figure 1c, we obtain the following answers for cube view $\Psi_{sum}[\texttt{Country}, \texttt{Day}]$:

$(\texttt{USA}, 1/1/12) \mapsto 0$, $(\texttt{Canada}, 1/1/12) \mapsto 10500$,
$(\texttt{England}, 1/1/12) \mapsto 10000$, $(\texttt{UK}, 1/1/12) \mapsto 10000$,
$(\texttt{USA}, 1/2/12) \mapsto 3000$, $(\texttt{Canada}, 1/2/12) \mapsto 5500$,
$(\texttt{England}, 1/2/12) \mapsto 1400$, $(\texttt{UK}, 1/2/12) \mapsto 1400$.

Combining these results with $\mathcal{R}^{\texttt{Continent}}_{\texttt{Country}}$, we will obtain cube view $\Psi_{sum}[\texttt{Continent}, \texttt{Date}]$ as:

$(\texttt{NA}, 1/1/12) \mapsto 10500$, $(\texttt{Europe}, 1/1/12) \mapsto 20000$,
$(\texttt{NA}, 1/2/12) \mapsto 8500$, $(\texttt{Europe}, 1/2/12) \mapsto 2800$,

which is not correct since sales for London have been counted twice for Europe. □

Example 6. For the EHM Location dimension of Figure 3, the summarizability sets are as follows:

$SumSets_{\text{Loc}}(\texttt{All}) = \{\{\texttt{Continent}\}, \{\texttt{Country}\}, \{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Continent}) = \{\{\texttt{Country}\}, \{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Country}) = \{\{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{State}) = \{\{\texttt{City}\}\}$,
$SumSets_{\text{Loc}}(\texttt{Province}) = \{\{\texttt{City}\}\}$.

As shown above, the summarizability sets of each category contain at least one set that is fully composed of its direct children. Therefore, this dimension is summarizable. □

In Proposition 1 below, we characterize the notion of summarizability in terms of roll-up relations (i.e. in purely algebraic terms). As shown by Example 4, in a compliant EHM dimension we can take advantage of subcategory roll-ups to summarize over existing cube views and compute values for higher level cube views. This justifies the presence of subcategories in the following proposition.

PROPOSITION 1. A compliant EHM instance $\mathcal{D}$ with schema $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$ is summarizable iff, for every category $c \in \mathcal{C}$ and base subcategory $b \in \mathcal{S}$, there exists a set of subcategories $\mathcal{T} = \{s_1, ..., s_n\}$, all children of subcategories in c, such that:

1. $dom(\mathcal{R}^c_b(\mathcal{D})) \subseteq dom(\bigcup_{s \in \mathcal{T}} \mathcal{R}^s_b(\mathcal{D}))$, and
2. The relation $\{(e_b, e_s) \mid e_b \in dom(\mathcal{R}^c_b(\mathcal{D}))$ and $(e_b, e_s) \in \bigcup_{s \in \mathcal{T}} \mathcal{R}^s_b(\mathcal{D})\}$ is a function. □

Remark 1. Proposition 1 can be applied without change to an HM instance. As before, we assume that every category in the schema has a single subcategory. □

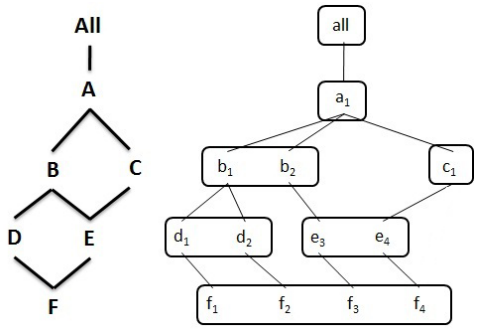


Figure 4: An HM dimension with non-transitive summarizability sets

Example 7. Consider the HM Location dimension of Figure 1. For category $c = \texttt{Continent}$ and base subcategory $b = \texttt{City}$, we have only one option as the set of child subcategories for category Continent, namely $\mathcal{ST} = \{\texttt{Country}\}$. Now, we have:

$$\begin{aligned} \mathcal{R}^c_b(\mathcal{D}) &= \{(\texttt{LA}, \texttt{NA}), (\texttt{Vancouver}, \texttt{NA}), (\texttt{Ottawa}, \texttt{NA}), \\ &\quad (\texttt{London}, \texttt{Europe})\}. \\ \bigcup_{s \in \mathcal{T}} \mathcal{R}^s_b(\mathcal{D}) &= \{(\texttt{LA}, \texttt{USA}), (\texttt{Vancouver}, \texttt{Canada}), \\ &\quad (\texttt{Ottawa}, \texttt{Canada}), (\texttt{London}, \texttt{England}), \\ &\quad (\texttt{London}, \texttt{UK})\}. \end{aligned}$$

As a result, $dom(\mathcal{R}^c_b(\mathcal{D})) \subseteq dom(\bigcup_{s \in \mathcal{T}} \mathcal{R}^s_b(\mathcal{D}))$ and the first condition of Proposition 1 is satisfied.
As for the second condition in Proposition 1, we can see that the relation {(LA, USA), (Vancouver, Canada), (Ottawa, Canada), (London, England), (London, UK)} is not a function, therefore this condition is not satisfied. □

4. SUMMARIZABILITY SETS

To efficiently answer an aggregate query using pre-computed cube views, we need to be able to decide which cube views produce correct results for the given query. This decision requires the computation of the summarizability sets for categories involved in the aggregate query. In this section, we will show that, unlike HM models, summarizability sets can be efficiently computed for compliant EHM models.

4.1 Computing summarizability sets in HM

To compute and generate summarizability sets for arbitrary HM models, both the dimension schema and instance have to be processed. This is because, with the HM model, we cannot say much about properties of the summarizability sets. This is due to the fact that in HM, the schema may not reflect the hierarchy of the dimension instance.

Example 8. Summarizability sets in HM do not necessarily have the property of transitivity. This is illustrated with the dimension shown in Figure 4. Here, $\{D, E\} \in SumSets(B)$ and $\{B, C\} \in SumSets(A)$ but it is not the case that $\{D, E, C\} \in SumSets(A)$. □

Actually, to generate the summarizability sets for an arbitrary HM model, we need to process the dimension instance. One way to do this is by checking the two conditions of Proposition 1 for every category and every subset of its children (let's call this *the naive algorithm*).

PROPOSITION 2. The naive algorithm for computing summarizability sets of an HM instance takes $O(k \times 2^k \times n \lg n)$ time where k is the size of the dimension schema (i.e. number of categories) and n is the size of the dimension instance (i.e. number of data elements). □

The dimension instance is usually very large in size and as a result such a computation becomes impractical. We do not have a practical way to decide whether an aggregation over pre-computed cube views for child categories will produce correct aggregate results for a cube view on a parent category (unless all dimensions are known to be summarizable and remain summarizable after any future updates which is usually not the case).

4.2 Computing summarizability sets in EHM

As stated earlier, the hierarchy of subcategories in a compliant EHM dimension reflects the hierarchy of the dimension instance. As a result, we can use this structure (which is smaller in size) to generate the summarizability sets. In the following, we will provide an algorithm that computes summarizability sets for a compliant EHM dimension by only processing its dimension schema (and not the instance itself).

We will first provide a characterization of local summarizability solely expressed in terms of elements of the dimension schema (Proposition 1). This characterization will be based on the notion of base-containment vectors defined next.

Definition 5. Let $\mathcal{D}$ be a compliant EHM instance with schema $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$ and base subcategories $s_1, ..., s_m$. The *base-containment vector*, $bcv : \mathcal{S} \cup \mathcal{C} \rightarrow \{0,1\}^m$, is a total function that assigns every subcategory $s \in \mathcal{S}$ (and every category $c \in \mathcal{C}$) to an m-ary boolean relation $(v_1, ..., v_m)$ where v_k is 1, iff $s_k \nearrow^* s$ (or in the case of a category, iff $\theta(s_k) \nearrow^* c$), otherwise v_k is 0. □

Example 9. Consider the EHM dimension of Figure 3. The base-containment vector of category `Continent` is $(1,1,1)$ because all base subcategories (i.e. `City1`, `City2` and `City3`) roll up to the `Continent` subcategory which in turn belongs to the `Continent` category. Also, for subcategories `Country1` and `Country2` the bcv value is $(1,1,1)$ and $(0,0,1)$, respectively. □

We can define the following operators for base-containment vectors of the same arity:

1. Operator $+$ is the usual add operation for two relations of the same arity. Note that the operands and the result have the same arity, but they are not necessarily boolean (i.e. they can contain items greater than 1). As usual, the extension of $+$ for more than two operands will be denoted by $\sum$.

2. Operator $\oplus$ is the boolean add operation for two boolean relations of the same arity. The result will also be a boolean relation. We denote the extension of $\oplus$ for more than two operands by $\sum_{\oplus}$.

3. Operator $\leq$ compares two relations of the same arity and succeeds only if all elements of the first relation are smaller or equal to all elements of the second relation.

In addition, we will use $(1, ..., 1)_m$ to denote an m-ary relation where all items are 1.

The following characterization of summarizability is a result of Proposition 1 for compliant EHM dimensions.

COROLLARY 1. Let $\mathcal{D}$ be a compliant EHM instance with schema $\langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$. Any category $c \in \mathcal{C}$ is $\mathcal{K}$-summarizable where $\mathcal{K} = \{c_1, ..., c_n\} \subseteq \mathcal{C}_{\mathcal{D}}$, iff there exists a set of subcategories $\mathcal{T} = \{s_1, ..., s_m\} \subseteq \mathcal{S}$ with $\theta(s_i) \in \mathcal{K}$ such that $bcv(c) \leq \sum_{s_k \in \mathcal{K}} bcv(s_k) \leq [1 \ ... \ 1]_{1 \times m}$. □

Example 10. (example 9 continued) The `Continent` category is summarizable from `Country` in the EHM dimension of Figure 3. This is because if we choose $\mathcal{T} = \{$`Country1`$\}$, we have $bcv($`Continent`$) = (1,1,1)$ and $bcv($`Country1`$) = (1,1,1)$ therefore $bcv($`Continent`$) \leq bcv($`Country1`$) \leq (1,1,1)$. □

Now, our algorithm for computing summarizability sets is composed of three simple steps:

1. First, it computes bcv values for base subcategories themselves. Notice that this is straightforward since every base subcategory has only one non-zero item in its bcv.

2. Then, in a bottom-up order, bcv values for other subcategories will be computed. Notice that we can compute the bcv for any subcategory, having already computed bcv values for its child subcategories (which explains the bottom-up order).

3. Finally, having all base-containment vectors, we can check Proposition 1 for any category c and all combinations of its lower level subcategories, to see if the respective lower level categories belong to the summarizability set of c.

We will first demonstrate these steps with an example.

Example 11. Consider again the EHM dimension of Figure 3.
Step 1: We have three base subcategories for which the bcv values are as follows:

$$bcv(\texttt{City1}) = (1,0,0), \quad bcv(\texttt{City2}) = (0,1,0),$$
$$bcv(\texttt{City3}) = (0,0,1).$$

Step 2: For all other subcategories the bcv will be initialized to all 0 elements. In a loop (until every subcategory is processed), we will choose parents of already processed subcategories and add (boolean add) to their bcv, the bcv value of the child subcategory. For example, we will choose base subcategory `City1` as an already processed subcategory, and find all its direct parents (in this case only subcategory `State`), and add $bcv($`City1`$)$ to $bcv($`State`$)$, therefore $bcv($`State`$)$ becomes $(0,0,0) \oplus (1,0,0) = (1,0,0)$. Notice that for a subcategory that has more than one child, such as `Country1`, the bcv values of all the children will be added gradually in the loop to finally obtain the correct value for $bcv($`Country1`$)$ which is $(1,1,1)$. At the end of this loop, the bcv values for all subcategories have been computed.

Step 3: Finally, for any category such as `Country`, we first compute the bcv value by the boolean addition of the bcv values of its subcategories. In this case $bcv($`Country`$) = (1,1,1) \oplus (0,0,1) = (1,1,1)$. Then, we find the set of subcategories that roll up to some subcategory in `Country`, in this case {`State`, `Province`, `City1`, `City2`, `City3`} and let $\mathcal{L}$ be the set of all possible subsets. In this case, $\mathcal{L}$ will contain 32 subsets including {`City1`, `City2`, `City3`}, {`State`, `Province`} and {`City1`, `City2`, `City3`, `State`}. Now, for each of these 32 subsets, we must check if the condition in Proposition 1 is satisfied. If so, the corresponding set of categories will be added to the summarizability set of `Country`. For example, $bcv($`City1`$) + bcv($`City2`$) + bcv($`City3`$) = (1,1,1)$ which satisfies the condition, so category `City` will be added. On the other hand, $bcv($`State`$) + bcv($`Province`$) = (1,1,0)$ which does not satisfy the condition. Also, notice that the third subset will not be added regardless of the condition, because it involves the set of categories {`City`, `State`} and some subset of this (in this

case {City}) has already been added to the summarizability set of Country. □

Algorithm Compute-SumSets($\mathfrak{S}$)
Input: $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta \rangle$, the schema of compliant EHM dimension $\mathcal{D}$.
Output: $SumSets_{\mathcal{D}}$, the summarizability sets of $\mathcal{D}$.
Initialize $WorkingQueue$ to an empty queue
// Step 1: Compute bcv for base subcategories
For base subcategories $s_1, ..., s_m$:
$bcv_{1k}(s_i) = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases}$
Push s_i into $WorkingQueue$
// Step 2: Compute bcv for other subcategories from bottom up
For every non-base subcategory s:
initialize $bcv(s)$ to $[0, ..., 0]_{1\times m}$
Loop until $WorkingQueue$ is empty
Pop s from $WorkingQueue$
If (s is not already processed)
For every s_p such that $s \nearrow s_p$
$bcv(s_p) = bcv(s_p) \oplus bcv(s)$
Push s_p into $WorkingQueue$
// Step 3: Check satisfaction of Proposition 1 for every combination
// of a category and its descendants
For every category $c \in \mathcal{C}$:
Let $bcv(c)$ be $\sum_{\oplus} bcv(s_k)$ where s_k is a subcategory of c
Initialize $SumSets_{\mathcal{D}}(c)$ to an empty set
Initialize $\mathcal{L}$ to a list containing subsets of $\mathcal{S}$ that all roll up to some subcategory in c
Sort $\mathcal{L}$ in the ascending order of number of categories involved in each subset
For every set of subcategories $\mathcal{T}$ in $\mathcal{L}$
Let $\mathcal{K}$ be $\{\theta(s_i) \mid s_i \in \mathcal{T}\}$
If no subset of K is already in $SumSets_{\mathcal{D}}(c)$
If $bcv(c) \leq \sum_{s_k \in \mathcal{T}} bcv(s_k) \leq (1, ..., 1)_m$
Add $\mathcal{K}$ to $SumSets_{\mathcal{D}}(c)$
Return $SumSets_{\mathcal{D}}$.

PROPOSITION 3. Algorithm Compute-SumSets for computing summarizability sets for all categories of a compliant EHM dimension takes $O(k^2 \times 2^k)$ time where k is the size of the dimension schema (i.e. number of subcategories). □

5. CREATING EHM DIMENSIONS

By imposing the compliance conditions on EHM dimensions, we are putting an additional overhead on the creation of dimensions. However, the result reduces the costs of computing summarizability sets. This trade-off is well justified by the importance of fast query processing versus updates in OLAP.

In the process of creating an EHM dimension, say a DWH, the user defines and specifies categories (not subcategories). Also, when posing queries, the user thinks in terms of categories. For this reason, subcategories will be defined and maintained by the underlying DWH management system, to keep track of the exact structure of data elements. The existence of subcategories can be transparent to the user.

5.1 Creating an EHM instance

We propose an incremental algorithm for inserting data links (with already defined parent elements) into a compliant EHM instance. The algorithm guarantees that the instance will remain compliant after the insertion. This will also make it possible to create compliant EHM dimensions from scratch. For this purpose, initially the dimension instance will contain only one element all that belongs to subcategory All. The instance will be populated incrementally by adding links one by one. The algorithm works as described below.

We start with a predefined set of categories in the dimension schema (defined by the DWH user), and with only one subcategory per category. For inserting a link such as (e_1, e_2), the parent element (in this case e_2) should already exist in the instance (this is not necessarily the case for the child element). If e_1 (i.e. the child side of the link) is a new element, the category should be specified as part of the input. One of the following two scenarios will happen:

1. If the new link matches the structure of the subcategory hierarchy (i.e. there is a subcategory s within the category of e_1 such that if we put e_1 in s, the instance will remain compliant), then we simply add the link and set its subcategory to be s. We will call this a *compliant addition*.

2. If we cannot find such a subcategory, then the subcategory hierarchy must be restructured by adding new subcategories. First, a new subcategory will be added to the category of e_1. If, by the new addition, the functionality of the subcategory roll-up relations becomes violated, we will also have to add a new subcategory to the category of e_2 (i.e. the parent). The new subcategory will contain e_2 as its sole element (notice that we also had the option of moving along with e_2 any combination of its siblings except the one causing the non-functionality, hence multiple EHM models can be created, but our algorithm chooses only one). In some special cases, multiple categories can be affected. This kind of link addition is called *restructuring addition*.

A compliant addition is more frequent and less time consuming. But, when we start populating a dimension instance, restructuring additions may occur until the subcategory hierarchy finds its final form (one that matches the external reality).

We will demonstrate these steps with an example. The complete algorithm with details can be found in the extended version [3].

Example 12. Consider the process of creating the EHM instance of Figure 3 from scratch. We can assume the initial schema shown in Figure 5a has been provided as input (by the DWH user, with an instance containing only element, all).

Inserting (NA, all), (USA, NA), (Canada, NA) and (California, USA) is consistent with the initial subcategory hierarchy. Therefore, these are compliant additions and no change in the schema is required. The result of inserting these links is shown in Figure 5a. Now, when adding link (LA, California), LA will have no parent in subcategory Province and there is no subcategory in category City that matches this element. Therefore, we will have to create a new subcategory, City1, in category City. Notice we may optionally remove the old subcategory City from category City, because it has no elements (but we do not have to). The result of this step is shown in Figure 5b.

Next, we may add links (England, Europe), (UK, Europe), (BC, Canada) and (ON, Canada), all consistent with the subcategory hierarchy. So, without any change in the schema, the resulting instance is shown in Figure 5c.

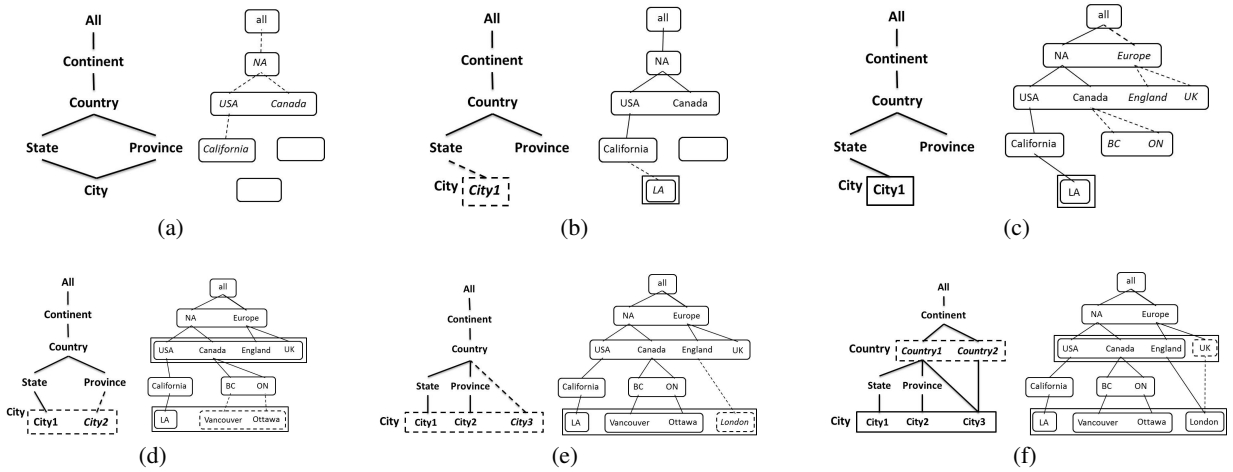


Figure 5: Steps of the add link algorithm for creating the `Location` dimension

The insertion of links (`Vancouver`, `BC`) and (`Ottawa`, `ON`) will require another restructuring of the subcategory hierarchy. In this case, as shown in Figure 5d, another subcategory, `City2`, must be added to category `City` that will reflect the structure of new elements being added.
Similarly, adding link (`London`, `England`) is a restructuring addition because no subcategory of `City` is connected only to `Country` (and not to `Province` or `State`). As shown in Figure 5e, `City3` will be added to category `City`.

Finally, when adding link (`London`, `UK`), the roll-up relation between subcategories `City3` and `Country` becomes a non-function, and as a result, we must split category `Country` into two subcategories, i.e. `Country1` and `Country2`, so that the instance becomes compliant again. The final result is shown in Figure 5f. □

5.2 EHM and ROLAP schemas

Multidimensional databases are commonly represented using the *star* or *snowflake* relational schemas. In the case of star, a fact table consisting of numerical measurements is directly joined to a set of dimension tables that contain descriptive attributes. On the other hand, the snowflake schema provides a hierarchical relational representation which is the normalized version of star [25].

The EHM model can be implemented by extending these ROLAP schemas and adding tables corresponding to categories that contain more than one subcategory (one column per subcategory). Figure 6a shows the representation of the `Location` dimension as a relational database instance for a star schema. Figure 6b shows tables that have been added to store subcategory information for categories `City` and `Country`. A similar extension can be applied to the snowflake schema.

With these relational extensions, the computation of roll-up relations between subcategories (e.g. $\mathcal{R}^{\texttt{Country1}}_{\texttt{City1}}$) is straightforward and requires only conditional joins with new tables. In addition, adding or deleting subcategories can be performed by simple add column or drop column operations. Data elements can be moved between subcategories using batch column updates.

6. EHM AND EXPRESSIVENESS

A natural question to ask is whether every HM dimension can be replaced by an equivalent compliant EHM dimension, i.e. one with the same elements and links in the dimension instance. We argue that the dimension instance is the representative (or model) of the outside reality being modeled. Therefore, expressiveness can be measured by the range of dimension instances a metamodel (such as HM or EHM) can capture. This is formalized by Definition 6. We first formalize what it means for two dimensions to be isomorphic. Notice that we do not bring the schema into the picture. So, the two dimensions can have different schemas (even more, one schema can be modeled using HM and the other using EHM).

In this section, we will be using the term dimension in a general sense to refer to HM dimensions as well as EHM dimensions that may or may not be compliant. We define the notion of expressiveness for classes of dimensions. For example, we refer to all the dimension instances that can be modeled using a compliant EHM dimension, as *the class of compliant EHM dimensions*.

Definition 6. (a) Dimension instances $\mathcal{D}_1 = \langle \mathcal{M}_1, \delta_1, <_1 \rangle$ and $\mathcal{D}_2 = \langle \mathcal{M}_2, \delta_2, <_1 \rangle$ (that may or may not be compliant) are *isomorphically equivalent*, iff there exists bijective function $f : \mathcal{M}_1 \to \mathcal{M}_2$ such that for arbitrary data elements $e_i, e_j \in \mathcal{M}_1$, it holds that $e_i <_1 e_j$ iff $f(e_i) <_2 f(e_j)$.
(b) A class of dimensions, $\mathcal{H}_1$, *expressively covers* $\mathcal{H}_2$, denoted $\mathcal{H}_2 \gg \mathcal{H}_1$, iff for any $\mathcal{D} \in \mathcal{H}_2$, there exists $\mathcal{D}' \in \mathcal{H}_1$ such that $\mathcal{D}$ and $\mathcal{D}'$ are isomorphically equivalent.
(c) $\mathcal{H}_1$ and $\mathcal{H}_2$ are *expressively equivalent*, iff $\mathcal{H}_1 \gg \mathcal{H}_2$ and $\mathcal{H}_2 \gg \mathcal{H}_1$.
(e) $\mathcal{H}_2$ is *more expressive* than $\mathcal{H}_1$, iff $\mathcal{H}_2 \gg \mathcal{H}_1$ but not $\mathcal{H}_1 \gg \mathcal{H}_2$. □

One can propose having one subcategory per category of an arbitrary HM dimension to produce such a complaint EHM dimension (with the same instance). Unfortunately, the result will not necessarily be a compliant instance. However, in [2] we show that by splitting categories in a correct fashion, we can make roll-up relations become total functions (a process that may result in multiple different base categories). A similar split process can be applied to subcategories of an EHM dimension (instead of categories) to make the instance compliant. The result is that every HM dimension can be replaced by a compliant EHM dimension without changing the dimension instance. This is expressed in Proposition 4.

City	State	Province	Country	Continent	All
LA	California	null	US	NA	all
Vancouver	null	BC	Canada	NA	all
Ottawa	null	ON	Canada	NA	all
London	null	null	England	Europe	all
London	null	null	UK	Europe	all

(a) Star representation of `Location`

City	City1	City2	City3
LA	1	0	0
Vancouver	0	1	0
Ottawa	0	1	0
London	0	0	1

Country	Country1	Country2
US	1	0
Canada	1	0
England	1	0
UK	0	1

(b) Storing EHM subcategories

Figure 6: Star implementation of the `Location` EHM dimension

PROPOSITION 4. The class of compliant EHM dimensions is expressively equivalent to the class of HM dimensions. □

As discussed before, elements in the summarizability sets of any dimension are key for speeding up OLAP queries. Another natural question is whether all these items are preserved after replacing an HM dimension with its EHM counterpart.

Definition 7. Let $\mathcal{D}$ and $\mathcal{D}'$ be isomorphically equivalent instances with schemas $\langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta\rangle$ and $\langle U', \mathcal{C}, \mathcal{S}', \nearrow', \theta'\rangle$ (i.e. with the same set of categories). We say that $\mathcal{D}'$ *covers summarizability items* of $\mathcal{D}$, denoted $\mathcal{D}' \geq_{sum} \mathcal{D}$, iff for every category $c \in \mathcal{C}$, having $\{c_1, ..., c_n\} \in SumSets_{\mathcal{D}}(c)$ implies that there exists $\mathcal{K} \in SumSets_{\mathcal{D}'}(c)$ such that $\mathcal{K} \subseteq \{c_1, ..., c_n\}$. □

We can show that not only summarizability items of HM dimensions can be preserved when replacing them by compliant EHM dimensions, but also in many cases we can add more items in the summarizability sets that did not exist before. This is captured in Proposition 5.

PROPOSITION 5. Let $\mathcal{D}_{HM}$ be an arbitrary HM dimension with schema $\mathfrak{S} = \langle U, \mathcal{C}, \mathcal{S}, \nearrow, \theta\rangle$. There exists an isomorphically equivalent compliant EHM dimension $\mathcal{D}_{EHM}$ with schema $\mathfrak{S}' = \langle U', \mathcal{C}, \mathcal{S}', \nearrow', \theta'\rangle$ such that $D_{EHM} \geq_{sum} D_{HM}$. □

Example 13. The EHM `Location` dimension covers summarizability items of the HM `Location` dimension of Figure 1. The inverse is not true since $\{\texttt{Country}\} \in SumSets_{Loc}(\texttt{Continent})$ for the EHM dimension but `Continent` is not summarizable from `Country` in the HM dimension. □

Now, a more important comparison should be made between the class of summarizable HM dimensions and the class of summarizable EHM dimensions. This is expressed in Corollary 2 which is a direct result of Proposition 5.

COROLLARY 2. The class of summarizable EHM dimensions is more expressive than the class of summarizable HM dimensions. □

7. EXPERIMENTS

Proposition 5 states that our extended model allows for modeling compliant EHM dimensions with summarizability sets that cover all the elements of the summarizability sets of their HM counterparts. This means that we can take advantage of more pre-computed views in answering higher level aggregate queries. We claim that this has a significant impact on the efficiency of MD query answering. In this section, we provide experimental support for this claim, comparing the HM vs. EHM implementation of an MD setting with three dimensions: `Branch`, `Date` and `Product`. HM schemas for these dimensions are shown in Figure 7.

For the EHM implementation, the schemas for `Date` and `Product` remains the same but the `Branch` schema will be different which is shown in Figure 8.

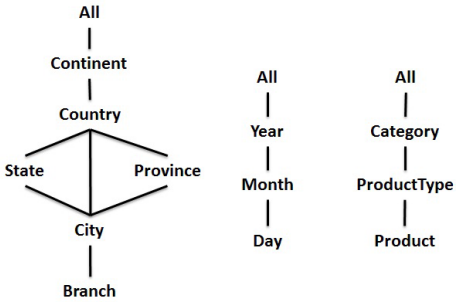


Figure 7: HM schemas for dimensions `Branch`, `Date`, `Product`

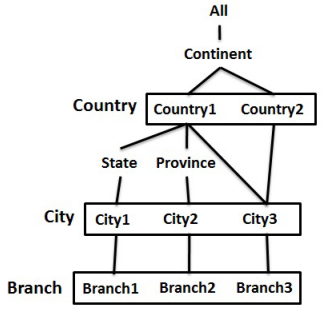


Figure 8: EHM version of the `Branch` dimension schema

Now, a star implementation of our HM dimensions will be composed of the following relational schemas (for simplicity and without loss of generality, we are assuming the fact tables contains only on measure, namely `Sales`):

$\texttt{BranchStar}(\texttt{Branch}, \texttt{City}, \texttt{State}, \texttt{Province}, \texttt{Country}, \texttt{Continent}, \texttt{All})$,
$\texttt{DateStar}(\texttt{Day}, \texttt{Month}, \texttt{Year}, \texttt{All})$,
$\texttt{ProductStar}(\texttt{Product}, \texttt{ProductType}, \texttt{Category}, \texttt{All})$,
$\texttt{FactTable}(\texttt{Branch}, \texttt{Day}, \texttt{Product}, \texttt{Sales})$

As explained in Section 5.2, for the EHM implementation of ROLAP relational schemas, we need additional tables corresponding to categories that contain more than one subcategory. In our experiment setting, only dimension `Branch` has categories with more than one subcategory (these categories are `Branch`, `City` and `Country`). Therefore, in a star implementation of our EHM dimensions, in addition to the above relations, we need the following relational schemas:

$\texttt{BranchSubcats}(\texttt{Branch}, \texttt{Branch1}, \texttt{Branch2}, \texttt{Branch3})$,
$\texttt{CitySubcats}(\texttt{City}, \texttt{City1}, \texttt{City2}, \texttt{City3})$,
$\texttt{CountrySubcats}(\texttt{Country}, \texttt{Country1}, \texttt{Country2})$

We consider the performance at aggregate query answering, and

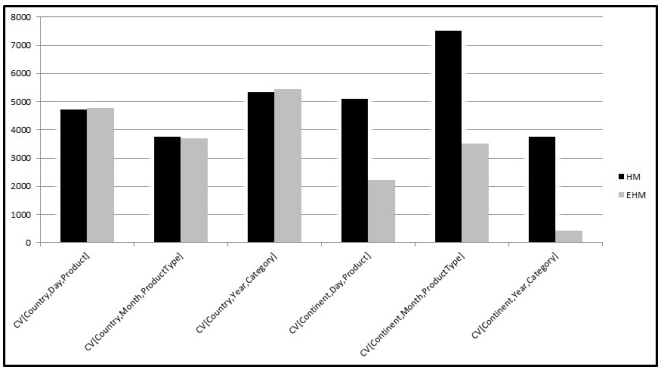


Figure 9: Comparison Between HM and EHM in aggregate query answering

we use IBM DB2 v9.7 for our experiments. We implemented a data generator in Java, to load a sizeable amount of test data into our MD databases for both HM and EHM. Starting with initial random data, the program generates elements for each category in the dimension schema, but taking into account the hierarchy levels. That is, the lower the category level the bigger the set of generated elements. For the `Branch` dimension, we inserted 30000, 2000, 400, 300, 200 and 5 distinguished data elements in categories `Branch`, `City`, `State`, `Province`, `Country` and `Continent`, respectively. For the `Date` dimension, we inserted 3650, 120 and 10 distinguished data elements in categories `Day`, `Month` and `Year`, respectively. Similarly, for the `Product` dimension, we inserted 2000, 50 and 10 distinguished data elements in categories `Product`, `ProductType` and `Category`, respectively. The fact table was populated with 1 million records.

In order to compare the query answering performance of the two models, we posed aggregate queries at different granularities for the three dimensions. For the `Branch` dimension we considered the level of categories `Country` and `Continent`. For dimensions `Date` and `Product`, we considered all levels except category `All`. We have chosen 6 combinations and provided the results in Figure 9. The horizontal axis shows the chosen granularity of each cube view and the vertical axis shows the average time in milliseconds it took the DBMS to compute the cube view (each query was posed 10 times and the average was computed).

With the EHM implementation, after cube views at the level of `Country` were computed, we could use them for more efficient computation of cube views at the level of `Continent`. This was not the case for the HM implementation since such reuse of pre-computed cube views would result in incorrect aggregate values.

As Figure 9 shows, for computing cube views at the level of category `Country`, the HM and EHM implementations yield similar results, but the difference in query answering time for computing results at the level of `Continent` is significant. Also, as we move to higher levels of all dimensions (in this case the level of [`Continent`, `Year`, `Category`]), the improvement in query answering time becomes much more apparent.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have addressed three important problems in relation to MD databases:

1. We have proposed and analyzed a new MD model which is an extension to the HM model with subcategories. This so-called *extended HM* (EHM) model has interesting properties with regard to summarizability. Namely, summarizable HM dimensions will still be summarizable in EHM without change. Additionally, in the case of non-summarizable HM dimensions, it allows for modeling dimensions in which more categories are summarizable from their lower level descendants. This in turn results in more efficient query answering by reusing pre-computed cube views at the level of those descendant categories.

2. One of the important issues in improving query answering performance for MD databases is to choose a set of cube views for materialization when it is too expensive to materialize all cube views [14]. Later, when another aggregate query needs to be computed, we need to be able to decide whether this new query can be computed using the set of materialized pre-computed cube views. To make this latter decision, we need an efficient algorithm that given a cube view can determine from which pre-computed cube views it can be computed. With the HM model such an algorithm will have to process the dimension instance which is usually very large and makes the computation impractical. With the help of the constraints that exist in the EHM model, we have designed such an algorithm for compliant EHM dimensions that runs in constant time with regard to the instance size.

3. We have provided a formal characterization of expressiveness for classes of dimension models. We have established that the class of summarizable HM dimensions is less expressive than that of summarizable EHM dimensions.

In addition, we have shown how EHM dimensions can be implemented using familiar ROLAP schemas with minor modifications to the underlying representation of tables.

1. There are many interesting research directions that are opened by this proposal:

2. Our naive algorithm for adding data links to an EHM instance was intended to demonstrate how an EHM model can be generated from scratch. We will extend our work by designing more efficient algorithms for insertions into an EHM dimension. Those algorithms can possibly insert a whole path starting from a base category to category `All` at once, rather than adding links in a path one by one. The optimization and implementation of this algorithm will be left for future work.

3. The EHM model allows for a schema-based repair approach that transforms non-summarizable HM dimensions into summarizable EHM dimensions. Such a transformation can be used as a replacement for both data and structural repairs without having to change the dimension instance or remove user-defined categories in the schema, thus resolving issues introduced by previous repair approaches.

4. In this paper, we have not considered consistent query answering for aggregate queries. The main problem is about doing CQA under summarizability constraints without explicitly computing all repairs. In this direction, we should investigate the existence of a corresponding notion of canonical instance introduced in [6], that was used to do and approximate consistent answers (under their repair semantics).

5. In [26] what-if or hypothetical queries in an OLAP setting have been investigated. These are classic OLAP (cube) queries but evaluated under assumed scenarios. It would be interesting to extend this work in an EHM setting.

6. An alternative to computing summarizability sets from scratch would be to compute them once and then check after any update,

to see if the sets require modification. Specially, in the case of HM dimensions where computing summarizability sets before queries is not efficient, this might be a viable solution that requires further investigation.

7. Our characterization of expressiveness was based on a notion of equivalence with regard to dimension instances. In [17], the authors define a notion of equivalence for schemas in the presence of dimension constraints where the focus is on information capacity (without considering the instance). This would result in a different characterization of expressiveness which is worth exploring.

8. Our relational schema extension for implementing ROLAP schemas can be formalized with the notion of schema evolution [12]. This can be done by providing a language of schema modification operators to concisely express schema changes. As a result the DBA will be allowed to evaluate the effects of such schema changes.

Acknowledgments: This research was funded by NSERC Discovery, an NSERC-IBM CRD, and the NSERC Strategic Network on Business Intelligence (BIN). L. Bertossi is a Faculty Fellow of IBM CAS.

9. REFERENCES

[1] M. Arenas, L. Bertossi and J. Chomicki. Consistent Query Answers in Inconsistent Databases. Proc. ACM PODS'99, 1999, pp. 68-79.

[2] S. Ariyan and L. Bertossi. Structural Repairs of Multidimensional Databases. Proc. Alberto Mendelzon International WS of Foundations of Data Management (AMW'11), *CEUR Workshop Proceedings*, Vol. 749, 2011.

[3] S. Ariyan and L. Bertossi. A Multidimensional Data Model with Subcategories for Flexibly Capturing Summarizability (extended version). `http://people.scs.carleton.ca/~bertossi/papers/ehmExt.pdf`.

[4] L. Bertossi. Consistent Query Answering in Databases. *ACM SIGMOD Record*, June 2006, 35(2):68-76.

[5] L. Bertossi. *Database Repairing and Consistent Query Answering*, Morgan & Claypool, Synthesis Lectures on Data Management, 2011.

[6] L. Bertossi, L. Bravo and M. Caniupan. Consistent Query Answering in Data Warehouses. In Proc. Alberto Mendelzon International Workshop on Foundations of Data Management (AMW'09), *CEUR Workshop Proceedings*, Vol. 450, 2009.

[7] L. Bravo, M. Caniupan and C. Hurtado. Logic Programs for Repairing Inconsistent Dimensions in Data Warehouses. Proc. of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW'10), *CEUR Workshop Proceedings*, Vol. 619, 2010.

[8] L. Cabibbo and R. Torlone. Querying Multidimensional Databases. Proc. DBLP'97, Springer LNCS 1369, 1997, pp. 319-335.

[9] M. Caniupan. Handling Inconsistencies in Data Warehouses. In *Current Trends in Database Technology*, Springer LNCS 3268, 2004, pp. 160-176.

[10] M. Caniupan and A. Vaisman. Repairing Dimension Hierarchies under Inconsistent Reclassification. In *Advances in Conceptual Modeling. Recent Developments and New Directions*, ER Workshops FP-UML, MoRE-BI, Onto-CoM, SeCoGIS, Variability@ER, WISM, Brussels, Belgium, 2011, pp. 75-85.

[11] M. Caniupan, Bravo, L. and Hurtado, C. Repairing Inconsistent Dimensions in Data Warehouses. To appear in *Data & Knowledge Engineering*. DOI: 10.1016/j.datak.2012.04.002, 2012.

[12] C. Curino, H. J. Moon and C. Zaniolo. Graceful Database Schema Evolution: The Prism Workbench. PVLDB, 2008, 1(1):761-772.

[13] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2000.

[14] V. Harinarayan, A. Rajaraman and J. Ullman. Implementing Data Cubes Efficiently. Proc. SIGMOD, 1996, pp. 205-216.

[15] C. Hurtado. Structurally Heterogeneous OLAP Dimensions. PhD Thesis, Computer Science Department, University of Toronto, 2002.

[16] C. Hurtado and C. Gutierrez. Computing Cube View Dependences in OLAP Datacubes. Proc. International Conference on Scientific and Statistical Database Management, IEEE CS Press, 2003, pp. 33-42.

[17] C. Hurtado and C. Gutierrez. Equivalence of Olap Dimension Schemas. In *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, Wilhelminenburg Castle, Austria, 2004, pp. 176-195.

[18] C. Hurtado and C. Gutierrez. Handling Structural Heterogeneity in OLAP. In *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, R. Wrembler and C. Koncilia (eds.), Idea Group, Inc. 2006.

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

[20] C. Hurtado and A. Mendelzon. Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. Proceedings ICDT'01, Springer LNCS 1973, 2001, pp. 375-389.

[21] C. Hurtado and A. Mendelzon. OLAP Dimension Constraints. Proceedings PODS'02, 2002, pp. 169-179.

[22] C. Hurtado, A. Mendelzon, and A. Vaisman. Updating OLAP Dimensions. Proc. International Workshop on Data Warehousing and OLAP (DOLAP), ACM Press, 1999, pp.60-66.

[23] C. Hurtado, A. Mendelzon and A. Vaisman. Maintaining Data Cubes under Dimension Updates. Proc. ICDE'99, IEEE CS Press, 1999, pp. 346-355.

[24] W. Inmon. *Building the Data Warehouse*. 4th Edition, Wiley, 2005.

[25] R. Kimball and M. Ross. *The Data Warehouse Toolkit: the Complete Guide to Dimensional Modeling*. Wiley, 2nd edition, April 2002.

[26] L. V. S. Lakshmanan A. Russakovsky and V. Sashikanth. What-if OLAP Queries with Changing Dimensions. Proc. ICDE'08, IEEE CS Press, 2008, pp. 1334-1336.

[27] H. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. Proc. International Conference on Scientific and Statistical Database Management, IEEE CS Press, 1997, pp. 132-143.

[28] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Extending Practical Pre-Aggregation in On-Line Analytical Processing. Proc. VLDB'99. Morgan Kaufmann, 1999, pp. 663-674.

[29] M. Yaghmaie, L. Bertossi, and S.Ariyan. Repair-Oriented Relational Schemas for Multidimensional Databases. Proc. EDBT'12, 2012, pp. 408-419.