Semantics-Based Multidimensional Query Over Sparse Data Marts

Claudia Diamantini, Domenico Potena, and Emanuele Storti(✉)

Dipartimento di Ingegneria dell'Informazione,
Università Politecnica delle Marche - Via Brecce Bianche, 60131 Ancona, Italy
{c.diamantini,d.potena,e.storti}@univpm.it

Abstract. Measurement of Performances Indicators (PIs) in highly distributed environments, especially in networked organisations, is particularly critical because of heterogeneity issues and sparsity of data. In this paper we present a semantics-based approach for dynamic calculation of PIs in the context of sparse distributed data marts. In particular, we propose to enrich the multidimensional model with the formal description of the structure of an indicator given in terms of its algebraic formula and aggregation function. Upon such a model, a set of reasoning-based functionalities are capable to mathematically manipulate formulas for dynamic aggregation of data and computation of indicators on-the-fly, by means of recursive application of rewriting rules based on logic programming.

Keywords: Logic-based formalisation of performance indicators · Multidimensional model · Sparse data marts · Query rewriting · Virtual enterprises

1 Introduction

More and more organisations recognise today the need to monitor their business performances to detect problems and proactively make strategic and tactical decisions. In this context, one of the biggest challenges is represented by consolidating performance data from disparate sources into a coherent system. Due to the lack of enforced integrity and relationship, Performance Indicators (PIs) based on this data are often incomplete, conflicting, or limited to a particular department/function within the organisation. Such a problem is more critical in a Virtual Enterprise (VE) environment, where the data providers can be very different. Heterogeneous organisations are in fact characterised by high levels of autonomy, that determines several kinds of data integration problems.

In the context of indicator management, such heterogeneities derive from internal business rules of the enterprise as well as the structure of the information system and hence cannot be simply resolved. Indeed, structural heterogeneity refers to the granularity level at which data are aggregated. For instance, some enterprises can have their revenues calculated globally for all its products,

S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 190–202, 2015.
DOI: 10.1007/978-3-319-22729-0_15

or disaggregated product by product. Structural heterogeneity also applies to the formula adopted to calculate an indicator: again, the total revenue can be given as unique number, or the enterprise can store the total sales and costs separately. As a consequence, the interaction between granularity and formula structural heterogeneities needs to be carefully managed. Moreover, after data reconciliation, in traditional data warehouses query answering is simply realised by aggregating over the available values, and this can generate unreliable results (e.g. obtaining a cost of 100 at the 1st semester 2013 and of 500 at the 2nd semester 2013 can be interpreted erroneously as a sudden increase of costs, while it turns out that the cost for the 1st semester has been calculated by summing over the first two months only). For such reasons we believe that awareness of the completeness of results is a fundamental feature of a supporting environment. This is especially true at VE level, where besides heterogeneities in dimension hierarchies and members, usually there is no agreement about which level of aggregation for data will be used, and also corporate policies about monitoring may change over time (e.g., a partner can change frequency for data storage from monthly to weekly basis at any time). As a consequence, sparsity comes from the fact that, even for the same indicator, data will likely refer to different members, levels or dimensions.

In this paper we address these problems by presenting a semantic-based approach for multidimensional query over various data marts, capable to cope with sparse data marts by dynamic calculation of indicators values. The work has been developed during the EU project BIVEE[1], aimed at supporting innovation in a VE environment, that is therefore the target scenario of the paper. In our approach, local data are described in global terms by means of a *semantic multidimensional model*, in which we extend the data cube model with description of the structure of an indicator in terms of its algebraic formula. The model allows to reconcile dimensional heterogeneities by referring to the same representation of dimensions. Moreover, having the semantics of formulas, we are able to resolve structural heterogeneities by defining paths among cubes that describe how to compute an indicator on the basis of others.

Queries are posed over the global model and rewritten over local cubes. To overcome the sparsity issue, we introduce a *completeness check* functionality, aimed to check the completeness of a result. In any case when a value is not complete (for instance if the query asks for a semester, but just three months are available), the completeness check determines alternative ways for its calculation, by exploiting two complementary expansion rules: besides the more traditional roll-up expansion, we introduce the indicator expansion. Through such a new operator, the value of an indicator at a given level can be calculated by other indicator values at the same level, through its formula. Moreover, through logic-based formula manipulation it is possible to derive non-explicitly defined formulas from others. This novel approach enables to (1) dynamically derive non-materialized data through expansion rules and (2) obtain an answer

[1] http://www.bivee.eu.

to a query if and only if either it is materialized or it is produced by complete aggregation, ensuring the quality of results as a by-product.

The rest of the work is structured as follows: in next Section we introduce a case study that is used along the paper. Section 3 is devoted to present the semantic multidimensional model while Sects. 4 and 5 respectively address the query rewriting mechanisms and the completeness check, together with some implementative details and computational aspects. In Sect. 6 we discuss some relevant related work in the Literature. Finally, Sect. 7 is devoted to draw conclusions and discuss future work.

2 Case Study

In this Section we present a case study that will be used as example in the paper. Let us consider two enterprises ACME1 and ACME2, each with a data mart and willing to join a Virtual Enteprise to cooperate for a certain project. In order to answer queries over the whole VE, their data should refer to the same schema. However, several structural heterogeneities exists. As for dimensions, in fact, ProductDim is missing in ACME2, and the others refer to different granularities (e.g., there are *Year* and *Semester* levels for TimeDim in ACME1, but only *Year* for ACME2). Finally, also the set of indicators are different: I1 only for ACME1, I2 and I3 for both. We also know that I1 is aggregated through SUM and is calculated by ACME1 as I2+I3. I2 and I3 have no formula; the first is aggregated through AVG and the second through SUM (Fig. 1).

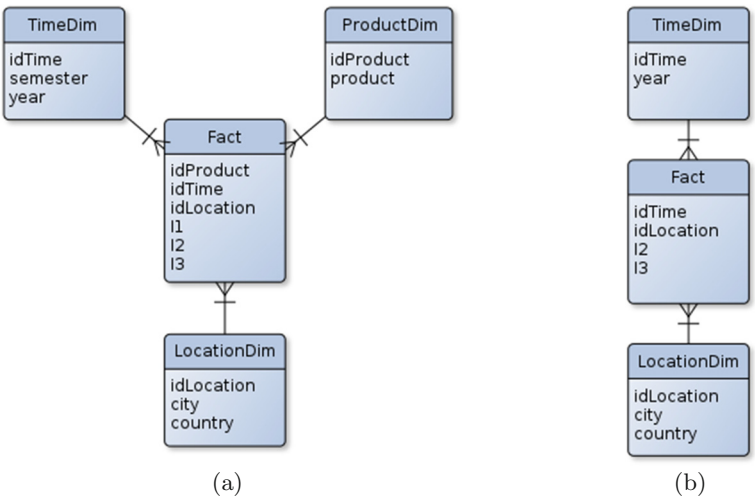

Fig. 1. Case study. The data marts for enterprises (a) ACME1 and (b) ACME2.

3 Semantic Multidimensional Model

The semantic multidimensional model is based on the formal representation of indicators and their formulas. We introduce here the basic notions of indicator, dimension and cube.

Indicator. An indicator is a way to measure a phenomenon and here it is defined as a pair $\langle aggr, f \rangle$, where $aggr$ is an aggregation function that represents how to summarize values of the indicator (e.g. sum, avg, min, max), while $f(ind_1, \ldots, ind_n)$ is a mathematical formula that describes the way the indicator is computed in terms of other indicators.

According to the classification adopted in data warehouse Literature, aggregation functions can be distributive, algebraic and holistic [1]. Indicator with a distributive aggregator can be directly computed on the basis of values at the next lower level of granularity, e.g. the Total Revenue for the 1st semester 2013 can be computed by summing values for the first six months of the 2013. Algebraic aggregators cannot be computed by means of values at next lower levels unless a set of other indicators are also provided, which transform the algebraic indicator in a distributive one; a classical example is given by the average aggregator (AVG), which must be computed on the basis of values at the lowest granularity level, whatever is the requested level. Indicators described by holistic functions (e.g. MEDIAN and MODE) can never be computed using values at next lower level. When no aggregation function is provided, indicators are computed by combining values of other indicators at the requested level.

We refer to [2,3] for further details about formulas and other properties of indicators.

Dimension. A Dimension is the coordinate/perspective along which the indicator is analysed (e.g., delivery time can be analysed with respect to means of transportation, the deliver and the date of delivery). Referring to the multidimensional model used in data warehouse [4], a dimension D is usually structured into a hierarchy of levels $L_1^D, \ldots, L_n^D$ where each level represents a different way of grouping members of the dimension (e.g., it is useful to group days by weeks, months and years). The domain of a level L_j^D is a set of members denoted by $\alpha(L_j^D)$, e.g. $\alpha(Country) = \{Portugal, Italy, Ireland, Greece, Spain, \ldots\}$.

Given $L_i^D \preceq L_j^D$, the transitive relation that maps a member of L_i^D to a member of L_j^D is defined as $partOf \subseteq \alpha(L_i^D) \times \alpha(L_j^D)$. This enables the possibility to move from a level to another with higher granularity through aggregation; this operation is called roll-up, the vice versa is the drill-down. To give some example, partOf("2013-01", "2013") means that the month "2013-01" is part of the year "2013", and partOf("Valencia", "Spain") that "Valencia" is part of "Spain". Although the fact that "Valencia is in Spain" is true in general, if an enterprise operating in Spain does not operate in Valencia, we should not consider (for this enterprise) values for the member "Valencia". Hence, we introduced a restriction $partOf_e$ of the $partOf$ relation, such that:

$$partOf(m_c, m_p) \wedge isValid(m_c, e) \rightarrow partOf_e(m_c, m_p),$$

where $isValid(m_c, e)$ is true if the member m_c makes sense for enterprise e. In this case each higher level member that is in $partOf$ relation with m_c is valid for e as well. More formally, $isValid(m_c, e) \wedge partOf(m_c, m_p) \rightarrow isValid(m_p, e)$.

The $partOf$ (hence $partOf_e$) relation is such that if $L_i^D \preceq L_j^D$, each member of L_i^D is in a $partOf$ relation with only one member of L_j^D; and each member of L_j^D is composed by at least one member of L_i^D. In other terms, the $partOf$ relation defines a partition of the members of level L_i^D.

As for the case study, let us assume for instance the following relations: $partOf_{A_1}(Barcelona, Spain)$, $partOf_{A_1}(Madrid, Spain)$, $partOf_{A_1}(Valencia, Spain)$ for ACME1 and only $partOf_{A_2}(Madrid, Spain)$ for ACME2.

Cube. According to the multidimensional model, each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions. We introduce this structure by the notion of *cubeElement*. A cubeElement is a storage element of the data mart for a given enterprise, and is defined as a tuple $ce = \langle ind \in I, m_1 \in \alpha(L_1^{D_1}), ..., m_n \in \alpha(L_j^{D_n}), e \in E, v \rangle$, where E is a set of enterprises and v is a value for ind.

A $cube(ind, e)$ is the logical grouping of all cubeElements provided by an enterprise e for an indicator ind, independently from the specific aggregation level adopted for the dimensions. We can assume the dimensional schema of a cube for ind (i.e., $ds(ind)$) to be the set of dimensions compatible with ind, e.g. $ds(I1) = \{TimeDim, ProductDim, LocationDim\}$. A cubeElement in $cube(ind, e)$ can be however defined over a subset of dimensions in the schema $ds(ind)$. In such a case, we assume that each missing dimension is aggregated at the highest level. Finally, the set of all cubes for a Virtual Enterprise is the global *data warehouse*.

4 Query Rewriting

Users query the global data warehouse for one or more indicators aggregated along the desired levels of the dimensions. Here we provide the definition of a multidimensional query and the query rewriting mechanism. For sake of simplicity, with no loss of generality the following definition refers to a single indicator.

Definition 1. A multidimensional query q over a Virtual Enterprise V is a tuple $\langle ind, W, K, V, \rho \rangle$, where:

- $ind \in I$ is the requested indicator,
- W is the set of levels $\{L_i^{D_1}, ..., L_j^{D_n}\}$ on which to aggregate,
- K is the collection of sets $K_i = \{m_{i1}, ..., m_{ik}\}$, $i = 1, ..., n$, of members on which to filter. K_i can be an empty set. In this case all members of the corresponding level are considered,
- $V = \{e_1, ..., e_n\} \subseteq E$

– ρ is *true* if the result is aggregated at the Virtual Enterprise level; otherwise the query will return a value for each enterprise in V.

The definition corresponds to the classical notion of aggregated OLAP query; in particular, W and ind are the elements of the target list, W is the desired roll-up level (or group-by component) while K allows slice and dice (suitable selections of the cube portion).

In our setting we face with a hybrid virtual-materialized data warehouse, since the aggregation levels of indicators are not the same for the different enterprises. In other terms, the query is posed on the global conceptual schema and needs to be rewritten in terms of the enterprise local schemas. Finally, the actual execution is realised over the enterprise logical schema[2]. We can rewrite a query for a given cube only if the dimensions requested in the query are a subset of the cube dimensions. Furthermore, not including a dimension D in the query means to request it at the highest level, that we conventionally denote by L_*^D.

Rule 1 *(Rewriting q Over a Cube).* Given a multidimensional query $q = \langle ind, W, K, V, \rho \rangle$, the multidimensional query q_c over the $cube(ind, e)$, with $e \in V$, is defined as $q_c = \langle ind, W', K', e \rangle$, where:

$$\begin{cases} (a) W' = W, K' = K & \text{if } W \text{ contains a level for each dimension in } cube(ind, e) \\ (b) W' = W \cup \{L_*^{D_x}\}, K' = K \cup \{ALL\} & \text{if } \exists \text{ a dimension } D_x \in cube(ind, e) \text{ that has no corresponding level in W} \end{cases}$$

Otherwise q_c cannot be defined.

Example. Given the following query:
$\langle I1, \{Year, Country\}, \{\{2013, 2014\}, \{Spain\}\}, \{ACME_1, ACME_2\}, false \rangle$,

it is rewritten as:
$q_1 = \langle I1, \{Product, Year, Country\}, \{\{ALL\}, \{2013, 2014\}, \{Spain\}\, ACME_1 \rangle$
$q_2 = \langle I1, \{Year, Country\}, \{\{2013, 2014\}, \{Spain\}\}, ACME_2 \rangle$.

The result of the query q_c is defined as the set $R(q_c)$ of tuples $\langle a, m_1, ..., m_n, e \rangle$ such that a is a value for ind, every m_i belongs to a different level, $e \in V$ and a cubeElement $\langle ind, m_1, ..., m_n, e, a \rangle$ is either (1) materialized in the data mart or (2) it can derived[3]. The following rules define how to derive a cubeElement from other cubeElements.

Rule 2 *(Roll-up Expansion).* Given $m_i \in \alpha(L_x^D), L_x^D \succeq L_y^D$, if a set of cubeElements $\langle ind, m_1, ..., m_j, ..., m_n, e, a_j \rangle$ exists such that $m_j \in \alpha(L_y^D)$ and $partOf_e(m_j, m_i)$, then the cubeElement $\langle ind, m_1, ..., m_i, ..., m_n, e, a \rangle$ can be computed, where $a = agg(a_j)$ denotes the aggregation over all elements a_j of the cubeElements under the aggregation function of the indicator ind.

[2] We do not focus on this step as it depends on the specific technology used for storage.
[3] It is straightforward to see that the result $R(q_c)$ of the query q_c, derived by applying the Rule 1 to the query q, is a subset of $R(q)$.

This rule corresponds to the classical roll-up OLAP operation in data warehouse systems. We hasten to note that for distributive aggregation function L_y^D can be any level lower than L_x^D, while for algebraic functions L_y^D must be the bottom level to assure the exact result.

Rule 3 *(Indicator expansion).* Let $ind = f(ind_1, ..., ind_k)$ be the formula defining ind in terms of indicators $ind_1, ..., ind_k$. If $\exists ce_i = \langle ind, m_1, ..., m_n, e, a_i \rangle, i = 1, ..., k$ then the cubeElement $\langle ind, m_1, ..., m_n, e, f(a_1, ..., a_k) \rangle$ can be computed.

This rule introduces a novel operation that allows us to infer new formulas not explicitly given, which is made possible by exploiting their formal representation. It also has to be noted that if a formula for ind was not specified in the model, it could be derived through mathematical manipulation, as described in a previous work [5], e.g. from $I1 = I2 + I3$ a formula for $I2$ can be computed as $I1 - I3$.

5 Query Completeness

The notion of answer completeness allows to define rules for the retrieval of data together with the evaluation of the completeness level of a query.

Definition 2 *(Completeness).* Given a query $q = \langle ind, W, K, V, \rho \rangle$, the result set $R(q)$ is complete if $R(q_c)$ is complete for each q_c obtained from the application of Rule 1.
Given a query $q_c = \langle ind, W', K', e \rangle$ with $K' = \{K_1, ..., K_n\}$, the result set $R(q_c)$ is complete iif:

- Condition (1) $|R(q_c)| = \prod_{j=1}^{n} |K_j|$,
- Condition (2) $\forall$ tuple $\langle a, m_1, ..., m_n, e \rangle \in R(q_c)$,
  - $\exists$ a cubeElement $ce = \langle ind, m_1, ..., m_n, e, a \rangle$ or
  - ce can be computed by applying a complete roll-up expansion or the indicator expansion rules.

Checking completeness of a multidimensional query over a cube (q_c) can be understood as (condition 1) a tuple-by-tuple calculus of the result set, one for each possible combination of elements of sets $K_1, ..., K_n$. Each tuple calculus corresponds (condition 2) either to the retrieval of the corresponding cubeElement (if the tuple is materialized) or to the execution of a set of suitable multidimensional queries for that tuple (in this case we speak of a virtual tuple), that can turn out to be either complete or not. As a consequence, the definition of query completeness is recursive. Furthermore, Rules 2 and 3 give us rewriting rules to define the form of queries to compute virtual tuples. Traversing the path of completeness check gives also as side effect a way to completely rewrite the original query.

5.1 Completeness Procedure

Completeness check is a procedure aimed at verifying the completeness of the result set for a given query at enterprise level. The procedure takes as input a query q at enterprise level, and produces as output a list of complete items and the list of incomplete items. The procedure is composed of the following macro-processes:

(1) Evaluation of results for the query q
(2) For each incomplete item t of the result:
  (2.1) Use of Rules 2 or 3 to determine alternative ways to obtain the item through a set of new queries $q_1, \ldots, q_n$
  (2.2) Recursive execution for each q_i.

In the following, each step is discussed together with an example. Let us consider a query $\langle I1, \{Year, Country\}, \{\{2013, 2014\}, \{Spain\}\}, \{ACME1, ACME2\}\rangle$. For lack of space, we show the execution of the procedure only on the data mart for ACME1, after the Rule 1 is exploited. Table 1 shows the cubeElements materialized in the highly sparse data mart for ACME1, in a tabular form for convenience of representation.

Table 1. Case study: the content of the data mart for ACME1.

Indicator	ProductDim	TimeDim	LocationDim	value
$I1$	ALL	2013 (Year)	Spain (Country)	90
$I2$	ALL	2014-S1 (Semester)	Spain (Country)	70
$I2$	ALL	2014-S2 (Semester)	Spain (Country)	66
$I3$	ALL	2014 (Year)	Barcelona (City)	25
$I3$	ALL	2014 (Year)	Madrid (City)	30
$I3$	ALL	2014-S1 (Semester)	Valencia (City)	10
$I3$	ALL	2014-S2 (Semester)	Valencia (City)	12

Step 1. According to the set K of the query, a table T is defined, including one placeholder for each possible item of the final result set. The header of the table is given by the set of levels W specified in the query, while rows are the cartesian product generated on K. At the beginning, each item is searched in the data mart. All the items that cannot be found are (temporarily) marked as incomplete.

Example. Table T includes two items: t_1 = ⟨I1,ALL,2013,Spain⟩ and t_2 = ⟨I1,ALL,2014,Spain⟩. t_1 is immediately available as added to the result, while t_2 is not found.

Step 2. If the list of incomplete items is empty (base case), then the query q is complete, and the execution ends by producing as output the results. Otherwise for each item t in the list of incomplete items, Rules 2 and 3 are used to derive t from other cubeElements in the data warehouse. Several alternative derivations can be possible for t, by applying roll-up over different dimensions (Rule 2) or by decomposing the indicator through different formulas (Rule 3). Hence, for each possible derivation a new set of queries $q_1, \ldots, q_n$ is constructed and checked for completeness, in a recursive fashion, by calling the procedure again. After the recursive call ends, if every query q_i returned a value, the item t is calculated according to the chosen rule. Otherwise, given that the chosen derivation could not obtain the value for t, the derivation is discarded, and the next possible derivation is applied, until no derivation is left. In such a case, if no derivation is found to be complete, t is definitevely marked as incomplete.

Example. t_2 is not found and the possible derivations are as follows. By executing Rule 2, drill-down can be used to expand either (a) Year to Semester or (b) Country to City levels. Product cannot be further expanded because it is the lowest level for ProductDim. By choosing[4] Rule 3, $I1$ is expanded into $I2$ and $I3$, given that its formula is $I2+I3$. Two new queries are then created:

- $q_1^{'} = \langle I_2, \{Product, Year, Country\}, \{\{ALL\}, \{2014\}, \{Spain\}\}, ACME1\rangle$
- $q_2 = \langle I_3, \{Product, Year, Country\}, \{\{ALL\}, \{2014\}, \{Spain\}\}, ACME1\rangle$.

No cubeElement is available for $q_1^{'}$. Let us suppose that Rule 2 is used in this case, drilling-down Year in Semester. As a result, three new queries are generated to retrieve I_2 for each semester of 2014 and for Spain. Results are available (i.e. 66 and 70), hence they are aggregated through the aggregation function of I_2 (AVG), and a result of 68 is obtained for $q_1^{'}$.

As for $q_2^{'}$, given that no cubeElements is available as well, Rule 2 is exploited. In this case drilling-down Year in Semester produces no result, so the derivation Country in City is tried. For the three cities in the model (Barcelona, Madrid and Valencia)[5], cubeElements are retrieved for the first two (with values 25 and 30) and only Valencia is missing. Recursively, a new derivation is tried for this incomplete item. This time, Rule 2 is executed to drill-down Year to Semester. Finally, results are available for both semesters, and values are aggregated through the aggregation function of I_3 (SUM), obtaining 12+10=22. $q_2^{'}$ is then computed as 25+30+22=77. By backtracking of the recursive procedure, as last step, t_2 is computed as I_2+I_3, i.e. 68+77=145.

5.2 Implementation and Computational Aspects

In order to define reasoning functionalities and support mathematical manipulation of PIs, the multidimensional model is represented in a logic-based language

[4] As the procedure explores the search space, if a solution exists, it is found whatever rule is chosen, although the order is critical w.r.t. execution time.

[5] As described in Sect. 3, in this drill-down for ACME1 we consider only the cities x such that the relations $partOf_{A_1}(x, Spain)$ hold in the data mart.

on which automatic inference can be exploited. To this end, we refer to Horn Logic Programming and in particular to Prolog, relying on XSB[6] (a logic programming and very efficient deductive database system) as reasoning engine.

While formulas are represented as facts, their manipulation as mathematical expressions is performed by a set of predicates. To this end, we have adapted and extended the Prolog Equation Solving System (PRESS) [6] that constitutes a first-order mathematical theory of algebra in LP, and allows for common algebraic manipulation of formulas. Among the offered functionalities, PRESS includes equation solving, equation simplification and rewriting, finding of inequalities and proving identities. Such predicates are useful to support more advanced reasoning functionalities including those needed to implement multidimensional query rewriting and completeness check.

This work can be framed in the general theory of query answering using views over incomplete data cubes. In the case of an implementation using only virtual views, its computational costs are dominated by the completeness check procedure. In the worst case, i.e. when no pre-aggregated data at intermediate levels exists, roll-up expansion has to traverse all the level hierarchy for every tuple of the result. This leads to a number of queries to be executed equal to $\frac{T(1-N^L)}{1-N}$, where T is the number of tuples in the database, N is the number of members of a level that are part of a member of the next higher level (e.g. three months are part of a quarter). To simplify the calculus, N is assumed to be constant for each level and member. Finally L is the number of levels in the hierarchy. Although in practice the worst case rarely occurs, it enlightens the value of materialization management.

In order to evaluate the costs related to execution times for completeness check, several steps ought to be considered. Here, we report the first results of the analysis of costs related to Rule 3, that is the novel type of rewriting proposed in this work. Conceptually, such an operation requires to find all possible rewritings for a formula and is independent on the size of data. We computed the execution time for all indicators in the model used for the BIVEE project [2], that includes 356 indicators where each of them has from 2 to 4 other indicators as operands (2.67 on average, only linear equations). The average execution time[7] in such a case is 735.60ms, ranging from a few ms to 2s.

6 Related Work

The sharing of data coming from distributed, autonomous and heterogeneous sources asks for methods to integrate them in a unified view, solving heterogeneity conflicts [7]. A common approach is to rely on a global schema to obtain an integrated and virtual view, either with a global-as-view (GAV) or a local-as-view (LAV) approach [8]. Query answering is one of the main tasks of a data integration system and involves a reformulation step, in which the query expressed

[6] http://xsb.sourceforge.net/.

[7] Experiments have been carried on a personal computer powered by an Intel Core i7-3630QM with 8 GB memory, running Linux Fedora 20.

by terms of the global schema is rewritten in terms of queries over the sources. According to the typology of the chosen approach (GAV or LAV), queries are answered differently, namely by unfolding or query rewriting based on views (see e.g. [9] in the context of distributed database systems, and [10,11] for data warehouse integration). Rewriting using views is used also for answering aggregate queries and as such can be related to the data warehouse field [11]. However, the multidimensional model has peculiarities that calls for specific studies. Heterogeneities hindering the integration of independent data warehouses may be classified on the basis of conflicts that occur at dimension or measure levels [12].

Recently, semantic representations of the multidimensional model have been proposed [13,14] mainly with the aim to reduce the gap between the high-level business view of indicators and the technical view of data cubes, to simplify and to automatise the main steps in design and analysis. In particular, in [15] a proprietary script language is exploited to define formulas, with the aim to support data cube design and analysis, while in [16] authors introduce an indicator ontology based on MathML, to define PI formulas in order to allow automatic linking of calculation definitions to specific data warehouse elements. Support for data cube design is considered in [15], while improvement of OLAP functionalities are presented in [17]. Both [13,14] tackle the problem of multidimensional modeling with semantics. The former relies on standard OWL-DL ontology and reason on it to check the multidimensional model and its summarizability, while the latter refers to Datalog inference to implement the abstract structure and semantics of multidimensional ontologies as rules and constraints. Although the complex nature of indicators is well-known, the compound nature of indicators is far less explored. Proposals in [11,15,16,18,19] include in the representations of indicators' properties some notion of formula in order to support the automatic generation of customised data marts, the calculation of indicators [15,18], or the interoperability of heterogeneous and autonomous data warehouses [11]. Anyway, in the above proposals, formula representation does not rely on logic-based languages, hence reasoning is limited to formula evaluation by ad-hoc modules.

Logic-based representations of dependencies among indicators are proposed in [19], although no manipulation of formulas is exploitable in this way.

7 Conclusion

In this paper we presented a semantics-based approach for dynamic calculation of performance indicators in the context of sparse distributed data marts. The methodology extends usual operators for query rewriting based on views, allowing to overcome structural heterogeneities among data mart schemas and making users aware of the completeness of the result of a query. Apart from the expressiveness of the reasoner, the approach does not set particular conditions on the types of formulas. At present our focus is on linear and polynomial formulas, since they represent the analytical definition of the 200 adopted indicators in the BIVEE framework. However, in general PRESS can deal with symbolic, trascendental and non-differential equations.

As extensions, while here we assume a unique aggregation function associated to each indicator, in future work we will consider different functions to aggregate data according to the dimension. For what concerns costs of computation, both a complete evaluation and the identification of methodological strategies to optimise completeness check exploitation (including pruning of search space and use of caching) will be discussed. Several strategies are possible for data materialization (e.g. lazy, eager or semi-eager), that is a classical issue in data warehouse implementation and will be discussed in a future work.

References

1. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Min. Knowl. Discov. **1**(1), 29–53 (1997)
2. Diamantini, C., Genga, L., Potena, D., Storti, E.: Collaborative building of an ontology of key performance indicators. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) OTM 2014. LNCS, vol. 8841, pp. 148–165. Springer, Heidelberg (2014)
3. Diamantini, C., Potena, D., Storti, E.: SemPI: a semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators. Future Generation Comput. Syst. (2015). http://dx.doi.org/10.1016/j.future.2015.04.011
4. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies, 1st edn. McGraw-Hill Inc, New York (2009)
5. Diamantini, C., Potena, D., Storti, E.: Extending drill-down through semantic reasoning on indicator formulas. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 57–68. Springer, Heidelberg (2014)
6. Sterling, L., Bundy, A., Byrd, L., O'Keefe, R., Silver, B.: Solving symbolic equations with press. J. Symb. Comput. **7**(1), 71–84 (1989)
7. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
8. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. PODS 2002, pp. 233–246. ACM, New York, NY, USA (2002)
9. Halevy, A.Y.: Answering queries using views: a survey. VLDB J. **10**(4), 270–294 (2001)
10. Cohen, S., Nutt, W., Sagiv, Y.: Rewriting queries with arbitrary aggregation functions using views. ACM Trans. Database Syst. **31**(2), 672–715 (2006)
11. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: Olap query reformulation in peer-to-peer data warehousing. Inf. Syst. **37**(5), 393–411 (2012)
12. Tseng, F.S., Chen, C.W.: Integrating heterogeneous data warehouses using xml technologies. J. Inf. Sci. **31**(3), 209–229 (2005)
13. Neumayr, B., Anderlik, S., Schrefl, M.: Towards Ontology-based OLAP: Datalog-based Reasoning over Multidimensional Ontologies. In: Proceedings of the Fifteenth International Workshop on Data Warehousing and OLAP, pp. 41–48 (2012)
14. Prat, N., Megdiche, I., Akoka, J.: Multidimensional models meet the semantic web: defining and reasoning on owl-dl ontologies for olap. In: Proceedings of the Fifteenth International Workshop on Data Warehousing and OLAP. DOLAP 2012, pp. 17–24. ACM, New York, NY, USA (2012)

15. Xie, G.T., Yang, Y., Liu, S., Qiu, Z., Pan, Y., Zhou, X.: EIAW: towards a business-friendly data warehouse using semantic web technologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 857–870. Springer, Heidelberg (2007)
16. Kehlenbeck, M., Breitner, M.H.: Ontology-based exchange and immediate application of business calculation definitions for online analytical processing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 298–311. Springer, Heidelberg (2009)
17. Priebe, T., Pernul, G.: Ontology-based integration of OLAP and information retrieval. In: Proceedings of DEXA Workshops, pp. 610–614 (2003)
18. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Softw. Syst. Model. **13**(3), 1015–1041 (2012)
19. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Inf. Syst. **35**(4), 505–527 (2010)