Extending Drill-Down through Semantic Reasoning on Indicator Formulas

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. Performance indicators are calculated by composition of more basic pieces of information, and/or aggregated along a number of different dimensions. The multidimensional model is not able to take into account the compound nature of an indicator. In this work, we propose a semantic multidimensional model in which indicators are formally described together with the mathematical formulas needed for their computation. By exploiting the formal representation of formulas an extended drill-down operator is defined, which is capable to expand an indicator into its components, enabling a novel mode of data exploration. Effectiveness and efficiency are briefly discussed on a prototype introduced as a proof-of concept.

1 Introduction

Performance measurement is the subject of extensive interdisciplinary research on information systems, organizational modeling and operation, decision support systems and computer science. Much work is devoted to categorize reference performance measures, or indicators [1, 2]. Strategic support information systems exploit results on data warehouse architectures. The multidimensional model has been introduced to suitably represent Performance Indicators (PI) and to enable flexible analyses by means of OLAP operators, facilitating managers in visualization, communication and reporting of PIs. Nevertheless, design and management of PIs are still hard. Among the major obstacles: (1) differences between the business view and the technical view of PIs, (2) information overload syndrome: managers are inclined to ask for more indicators than those actually needed [3], (3) interpretation of the meaning of indicators and their values. To some extent these obstacles relate to the fact that indicators are complex data with an aggregate and/or compound nature, as their values are calculated by applying some *formulas* defined over other indicators, or by aggregating raw data values, or both. Unawareness of the dependencies among indicators leads people to treat indicators as independent pieces of information, and is a cause of the information overload syndrome. Similarly, many disputes during managerial meetings come from a lack of a common understanding of indicators. To give an example, somebody states that the amount of investment in higher education in Italy is too low (far below the EU average), while somebody else states it is too high and should be lowered. Digging into this apparent contradiction, one discovers that both evaluate the amount of investment as the ratio between "total expenditure" and "student population", but whereas the former defines the "student population" as people officially enrolled in a course, the latter subtracts students who do not actually take exams. In order to fully grasp the meaning of the indicator an analysis of the way it is calculated is necessary. Furthermore, a correct interpretation of the value and trend of the indicator is unachievable without analysing its components.

Although the complex nature of indicators is well-known, it is not fully captured in existing models. The multidimensional model takes into account the aggregative aspect, defining a data cube as a multi-level, multidimensional database with aggregate data at multiple granularities [4]. The definition of powerful OLAP operators like drill-down directly comes from this model. Semantic representations of the multidimensional model have been recently proposed [4–8] 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 automatize the main steps in design and analysis.

The compound nature of indicators is far less explored. Proposals in [2, 9–14] include in the representations of indicators' properties some notion of formula in order to support the automatic generation of customized data marts, the calculation of indicators [9, 10, 13], or the interoperability of heterogeneous and autonomous data warehouses [12]. In the above proposals, formula representation does not rely on logic-based languages, hence reasoning is limited to formula evaluation by ad-hoc modules. No inference mechanism and formula manipulation is enabled. Formal, logic-based representations of dependencies among indicators are proposed in [2,14]. These properties are represented by logical predicates (e.g. *isCalculated* [14], *correlated* [2]) and reasoning allows to infer implicit dependencies among indicators useful for organization modeling, design, as well as reuse, exchange and alignment of business knowledge. An ontological representation of indicator's formulas is proposed in [11] in order to exchange business calculation definitions and to infer their availability on a given data mart through semantic reasoning, with strong similarities with our previous work [15].

In the present paper we propose to extend the data cube model with the description of the structure of an indicator given in terms of its algebraic formula. Relations between the aggregation function and the formula are taken into account. As the traditional multidimensional model leads to the definition of the drill-down operator, so the novel structure enables the definition of a novel operator we call *indicator drill-down*. Like the usual drill-down, this indicator increases the detail of a measure of the data cube, but instead of disaggregating along the levels of dimensions, it expands an indicator into its components. The two notions of drill-down are integrated thus allowing a novel two-dimensional way of exploring data cubes. To the best of our knowledge this is the first time that such an extended drill-down is considered. As a further contribution, we introduce a first-order logic theory for the representation of indicators' formulas and

Table 1. An excerpt of the enterprise glossary

Indicator	Description	Aggr	Formula
AvgCostsExtIdeas	Average costs of ideas produced by external users through the crowdsourcing platform	n/a	$\frac{CrowdInv}{NumExternalIdeas}$
CrowdInv	Total Investments for crowdsourcing activities: management of the platform, promoting and facilitating the participation,...	SUM	
IdeasAcceptedEng	The number of ideas accepted for engineering. These ideas are intended for future production	SUM	
IdeasProposed	Number of (internal/external) ideas proposed	SUM	$NumInternalIdeas+ NumExternalIdeas$
IdeaYield	Ratio of proposed ideas to ideas that have entered the engineering phase and will be developed	n/a	$\frac{IdeasAcceptedEng}{IdeasProposed}$
NumExternalIdeas	Number of ideas generated from the external stakeholders	SUM	
NumInternalIdeas	Number of ideas generated from enterprise's employees	SUM	

manipulation based on equation resolution. The set of predicates defines a knowledge base for reference, domain-independent indicators specification. Other predicates are introduced to define members' roll-up along dimensional hierarchies, as well as to state the set of indicators actually implemented in a given data mart, thus providing the specification for a certain application domain. The theory enables formula manipulation thus providing the full drill-down functionalities, allowing to expand an indicator into its components even if some are not explicitly stored in the data mart. Besides enabling the definition of a novel operator, the proposed logic representation extends the state of the art in the following ways: (1) we are not limited to the reference indicator specification, since equivalent definitions can be inferred, hence the evaluation of the formula can follow several paths, (2) indicators that are not explicitly stored in the data mart can be calculated by exploiting their relationships with other indicators represented in the knowledge base, (3) relevant relationships among indicators, like (inverse) dependency, (inverse) correlation, causality, influence can be inferred, while they have to be explicitly introduced with other approaches [2,13,14].

The rest of the paper is organized as follows: Section 2 introduces the case study that will be used as an illustrative example through the paper. Section 3 presents the proposed model, then Section 4 discusses its application to the definition of the extended drill-down operator. In Section 5 the proposal is evaluated. Finally Section 6 draws some conclusion and discusses future work.

2 Case Study

The present work is conceived within the EU Project BIVEE[1]. In this Section we present a case study that is based on the data mart (DM) used by one of the end-users of the project, and will be used as an illustrative example through the paper. In particular, we refer to an enterprise that develops innovative solutions,

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

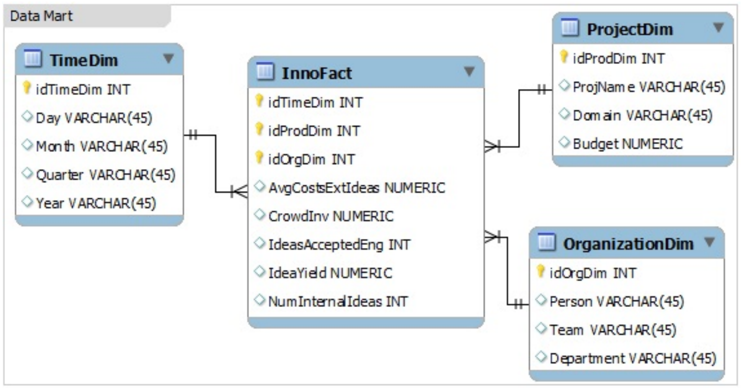


Fig. 1. The Innovation data mart

from metrology to robotics domains, to satisfy specific requests of customers. The enterprise adopts the open innovation paradigm [16], where innovation extends beyond the boundaries of the enterprise, involving both partners and customers as sources of innovative ideas. To this end, the enterprise hosts and manages a crowdsourcing platform. Finally, in the enterprise the work is organized on a project basis. Figure 1 shows the data mart used by the enterprise to analyse innovation projects. The data mart provides 5 indicators (i.e. measures in data warehouse terminology) and 3 dimensions, which represent different perspectives of analysis: time, organization and project. The attribute *budget* in the project dimension is not a level, but an informative attribute of a project. The description of measures is given in Table 1. In particular note that together with a textual description, the mathematical formula to calculate the indicator is provided for some indicators. When the formula does not exist, the indicator is said *atomic*, and is independent on other indicators. Otherwise the indicator is said *compound*, and the operands occurring in its formula are called *dependencies*. Note that compound indicators can be defined in terms of other compound indicators, producing a tree of dependencies, although it cannot be fully grasped in an informal glossary like the one presented. Also, aggregation functions are specified when applicable. When an aggregation function exists, it means that it is applied to the result of the formula in order to aggregate over dimensions. According to the case study and for the sake of simplicity, hereafter it is assumed that the same aggregation function applies to each dimension.

3 Semantic Multidimensional Model

The semantic multidimensional model is based on a first-order logic representation of indicators' formulas and their properties as well as of a multidimensional structure. Hence we first introduce the basic notions related to the multidimensional model.

Definition 1. *(Cube schema)* A cube schema S is a pair $\langle \mathcal{D}, I\rangle$, where I is a set of indicators $\{ind_1, \ldots, ind_m\}$ and $\mathcal{D}$ is the set of dimensions for I.

A dimension $D_i \in \mathcal{D}$ is the hierarchy of levels $L_1^{D_i} \preceq ... \preceq L_l^{D_i}$ along which measures are aggregated. The partial order $\preceq$ is such that if $L_1^D \preceq L_2^D$ then L_1^D rolls up to L_2^D (and L_2^D drills down to L_1^D). The domain of a level $L_j^{D_i}$ is a set of members $\{m_{j_1}, \ldots, m_{j_n}\}$ and will be denoted by $\alpha(L_j^{D_i})$, e.g. $\alpha(\text{Department}) = \{\text{RnD, RforI, ElectricDept, MechanicDept, ...}\}$. Referring to the Organization dimension of the case study, the following hierarchy holds: Person $\preceq$ Team $\preceq$ Department. In general the definition allows also for multiple hierarchies.

Definition 2. *(Cube element)* A cube element *ce* for the cube schema $S = \langle \mathcal{D}, I\rangle$ is the tuple $\langle m_1, \ldots, m_n, v_1, \ldots, v_m\rangle$ where each member belongs to a level of a dimension in $\mathcal{D}$ (i.e., $\forall m_i\ \exists {L_j}^{D_i}$ such that $m_i \in \alpha({L_j}^{D_i})$), and $\{v_1, \ldots, v_m\}$ are values for I.

In the following, we will not assume *completeness* of data cubes, i.e. cubes that include a cube element for any possible combination of members.

Central to our model is the notion of indicator: while in the standard cube schema definition (1) indicators are just labels, in the proposed model the structure of an indicator is taken into account.

Definition 3. *(Indicator)* An indicator *ind* $\in I$ is defined by the pair $\langle aggr, f\rangle$, where:

– $aggr \in \{\text{SUM}, \text{MIN}, \text{MAX}, \text{AVG}, \text{VAR}, \text{COUNT}, \text{NONE}, \ldots\}$ is an aggregation function that represents how to group values of the indicator;
– $f(ind_1, \ldots, ind_n)$ is the formula of the indicator, i.e. a mathematical expression that describes the way *ind* is computed in terms of other indicators $(ind_1, \ldots, ind_n) \in I$.

The label NONE is used to denote the absence of aggregation function, e.g. $IdeaYield = \langle NONE, IdeasAcceptedEng/IdeasProposed\rangle$. According to widely accepted models (e.g. [17]), aggregation is categorized in distributive, algebraic or holistic. Indicators with a distributive aggregator can be directly computed on the basis of values at the next lower level of granularity (e.g., SUM, MIN, MAX). Algebraic aggregator cannot be computed by means of values at next lower level 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). Indicators described by holistic functions can never be computed using values at next lower level (e.g., MEDIAN and MODE).

A formula is said to be *additive* if it includes only summation and differences of indicators, e.g. $f(ind_x, ind_y) = ind_x + ind_y$. Additivity is a relevant property since indicators with additive formulas and distributive aggregation functions (e.g., SUM) define a special subclass of indicators, for which holds that:

$aggr(f(ind_1, \ldots, ind_n)) = f(aggr(ind_1), \ldots, aggr(ind_n))$. This is not true in the general case, e.g. $AVG(x/y) \neq AVG(x)/AVG(y)$.

Indicators and their properties are represented by first-order logic predicates. We refer to Horn Logic Programming, and specifically to Prolog, as the representation language. $\texttt{formula}(ind, f)$ is a fact representing the formula related to an indicator. In the predicate *ind* is an indicator label, while f is an expression including algebraic operators like sum, difference, product, power, and operands are indicators' labels. Additivity is expressed by the predicate $\texttt{isAdditive}(ind)$. The set of formulas and its properties define a reference knowledge base for indicators specification. A further predicate $\texttt{hasInd}(c, ind)$ allows to state the presence of the indicator *ind* in the schema of the cube c. For what concerns the multidimensional structure, similarly to [18], the predicate $\texttt{rollup}(X, Y)$ is introduced to assert that a member X is mapped to the member Y of the next higher level to perform the roll-up operation.

Given the set of facts, rules are devised to implement reasoning functionalities. For instance the following rule implements the transitive closure of roll-up:

$\texttt{partOf}(X, Y) :- \texttt{rollup}(X, Y).$
$\texttt{partOf}(X, Y) :- \texttt{rollup}(X, Z), \texttt{partOf}(Z, Y).$

While formulas are represented as facts, manipulation of mathematical expressions is performed by specific predicates from PRESS (PRolog Equation Solving System), which is a formalization of algebra in Logic Programming for solving equations. Such predicates implement axioms of a first-order mathematical theory, and can manipulate an equation to achieve a specific syntactic effect (e.g., to reduce the number of occurrence of a given variable in an equation) through a set of rewriting rules. PRESS works in the domain of R-elementary equations, that is on equations involving polynomials, and exponential, logarithmic, trigonometric, hyperbolic, inverse trigonometric and inverse hyperbolic functions over the real numbers, although all indicators found in the analysis of real-world scenarios until now have linear formulas. PRESS is demonstrated to always find a solution for linear equations.

Due to lack of space, we cannot go into the details of the rule system. We just enlighten that the use of PRESS here allows to derive a new formula for an indicator that is not explicitly given. Among all possible inferred formulas, we are able to individuate the subset that can be actually calculated on the given data cube by the predicate $\texttt{hasInd}(c, \_)$. In the following we will refer to this high-level reasoning functionality as $\mathcal{F}(ind, C)$. Referring to the cube DM of the case study, the following formulas are derived by PRESS:

$IdeasProposed = NumInternalIdeas + NumExternalIdeas;$
$IdeasProposed = NumInternalIdeas + \frac{CrowdInv}{AvgCostsExtIdeas};$
$IdeasProposed = \frac{IdeasAcceptedEng}{IdeaYield}.$

$\mathcal{F}(IdeasProposed, DM)$ returns only the last two since, although the first one is the definition of the indicator provided in the knowledge base, it cannot be calculated in this way on DM due to the lack of $NumExternalIdeas$.

The formal representation and manipulation of the structure of a formula enables advanced functionalities like the definition of an extended drill-down operator, described in the next Section.

4 Extended Drill-Down

The present Section discusses how to exploit reasoning capabilities over indicator's formula by introducing a novel *indicator* drill-down operator. Like the usual drill-down, it increases the detail of a measure of the data cube, but instead of disaggregating along the levels of dimensions, it expands an indicator into its components. As the traditional drill-down is enabled by the notion of dimension's hierarchies, so the indicator drill-down arises from introducing the indicator's structure in the model. Furthermore, by reasoning on the logic representation proposed, the operator is able to extract values even for indicators not explicitly stored in the cube. We hasten to note that the rules defined to this end must work jointly on the structure of an indicator and on the structure of its dimensions. This integration of the notion of indicator in the multidimensional model is what enables the definition of the extended drill-down as the composition of the classic drill-down and of the indicator drill-down defined as follows.

Definition 4. *(Indicator drill-down)*
Given a schema S=$\langle\mathcal{D}$, $\{ind_1, \dots, ind_i, \dots, ind_n\}\rangle$ and an indicator ind_i with a formula $f_{ind_i} = f(ind_{i_1}, \dots, ind_{i_k})$, the indicator drill-down on ind_i is a function that maps a cube with schema S in a cube with schema S' such that:

– $S' = \langle\mathcal{D}, I'\rangle, I' = (I \setminus \{ind_i\}) \cup \{ind_{i_1}, \dots, ind_{i_k}\}$;
– instances are cube elements $ce = \langle m_1, \dots, m_h,\ v_1, \dots, v_{i_1}, \dots, v_{i_k}, \dots, v_n\rangle$, such that v_j is the value of the $j-$th indicator, m_p is the member of Dimension D_p and $v_i = f(v_{i_1}, \dots, v_{i_k})$.

Operationally, this means to access the definition of ind_i, extract the dependencies from its formula, and extract values for dependencies in order to build the new cube. This can be expressed as the rewriting of the multidimensional query generating the cube S.

Definition 5. *(Multi-dimensional Query)* A multi-dimensional query MDQ on a cube C with schema $\langle\mathcal{D}, I\rangle$ is a tuple $\langle\delta, \{ind_1, \dots, ind_m\}, W, K, \sigma\rangle$, where:

– δ is a boolean value introduced here to make explicit how the query is performed. If $\delta = false$ then indicator values are materialized in the cube, otherwise they are virtual, hence we assume they are calculated by aggregation of values at the lowest levels of dimensional hierarchies;
– $\{ind_1, \dots, ind_m\}$ is the set of requested indicators;
– W is the set of levels $\{L^{D_1}, ..., L^{D_n}\}$ on which to aggregate, such that $L^{D_i} \in D_i$ and $\{D_1, \dots, D_n\} \subseteq \mathcal{D}$;
– K is the collection of sets $K_h = \{m_{h_1}, ..., m_{h_k}\}$, $h := 1, ..., n$, of members on which to filter, such that each m_{h_j} belongs to $\alpha(L^{D_h})$. K_h can be an empty set. In this case all members of the corresponding level are considered;
– σ is an optional boolean condition on indicators' values.

$\{ind_1, \ldots, ind_m\}$ are the elements of the target list, W is the desired roll-up level (or group-by components) for each dimension, while K allows slice and dice (suitable selections of the cube portion). While K works on members, the filter σ defines a condition on other elements of the DM: both descriptive attributes of dimensional schema (e.g. $Budget$>50K) and values of indicators (e.g. $NumberInternalIdeas < NumberExternalIdeas$).

The result of a MDQ is a subset of the original cube where cube elements are $ce = \langle m_1, \ldots, m_n, v_1, \ldots, v_m \rangle$, where $\langle m_1, \ldots, m_n \rangle \in K_1 \times \ldots \times K_n$ and v_i is a value of the indicator ind_i. Given the notion of query, the drill-down can be seen as a rewriting of the original query $MDQ = \langle \delta, I, W, K, \sigma \rangle$ as $MDQ' = \langle \delta, I', W, K, \sigma \rangle$. Rewriting an indicator as its direct dependencies produces a correct query only if the data cube has been designed to store the set of indicators $\{ind_{i_1}, \ldots, ind_{i_k}\}$. The rewriting rules allowing to correctly specify the indicator drill-down are discussed in the following. They depend on the typology of formula and aggregation function. For the sake of simplicity we consider multidimensional queries with only one indicator in the target list.

Indicator Drill-Down Rule. Let $MDQ = \langle \delta, \{ind\}, W, K, \sigma \rangle$ be a query over the cube C with schema $\langle D, I \rangle$, where $ind = \langle aggr, f(ind_1, \ldots, ind_k) \rangle$. The indicator drill-down of MDQ is $MDQ' = \langle \delta, \{ind_1, \ldots, ind_k\}, W, K, \sigma \rangle$ where $\forall ind_i$ either $ind_i \in I$ or one of the following equivalence rules applies.

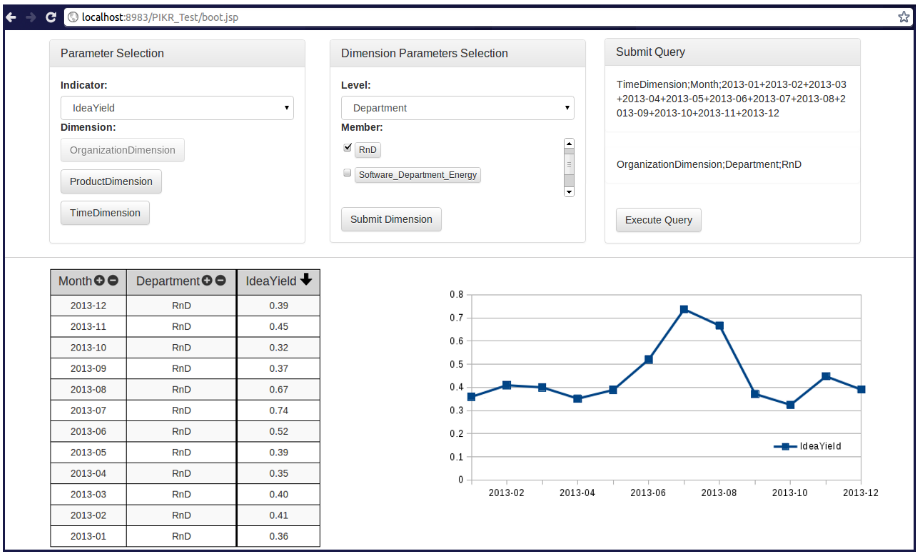
Equivalence Rules. Let $MDQ = \langle false, \{ind\}, W, K, \sigma \rangle$ be a query over the cube C with schema $\langle \mathcal{D}, I \rangle$, where $ind = \langle aggr, f \rangle$

– if ($aggr$ is distributive AND f is additive) OR ($aggr = NONE$):
  $MDQ = \langle false, g, W, K, \sigma \rangle$, $g \in \mathcal{F}(ind, C)$
– else: $MDQ = \langle true, g, W, K, \sigma \rangle$, $g \in \mathcal{F}(ind, C)$

The equivalence rules make use of the inference mechanism represented by $\mathcal{F}(ind, C)$, which defines any formula equivalent to f that can be inferred and is computed by indicators of the cube. The rule described in the else case captures the fact that for general aggregation functions the correct value of an indicator at a given level can be only obtained by calculating the formula at the lowest level of granularity (given that $\delta = true$), and then applying the aggregation on the resulting values. The first rule accounts for the commutativity property stated in the previous section that allows to apply the formula directly on the requested aggregation levels. This rule can be easily extended to algebraic aggregators given the well-known relation with distributive aggregators. For instance, in the case of the AVG the query becomes $MDQ = \langle \delta, \{\frac{ind'}{CountM}\}, W, K, \sigma \rangle$, where $ind' = \langle SUM, f \rangle$ and $CountM$ is a special function which returns the number of members m_0 of the lowest level such that $\texttt{partOf}(m_0, m), m \in K$.

5 Evaluation

A prototype of the system has been implemented as a proof-of-concept. This is part of a system offering additional services developed within the BIVEE



localhost:8983/PIKR_Test/boot.jsp

Parameter Selection

Indicator: IdeaYield

Dimension: OrganizationDimension | ProductDimension | TimeDimension

Dimension Parameters Selection

Level: Department

Member: ☑ RnD ☐ Software_Department_Energy

Submit Dimension

Submit Query

TimeDimension;Month;2013-01+2013-02+2013-03+2013-04+2013-05+2013-06+2013-07+2013-08+2013-09+2013-10+2013-11+2013-12

OrganizationDimension;Department;RnD

Execute Query

Month	Department	IdeaYield
2013-12	RnD	0.39
2013-11	RnD	0.45
2013-10	RnD	0.32
2013-09	RnD	0.37
2013-08	RnD	0.67
2013-07	RnD	0.74
2013-06	RnD	0.52
2013-05	RnD	0.39
2013-04	RnD	0.35
2013-03	RnD	0.40
2013-02	RnD	0.41
2013-01	RnD	0.36

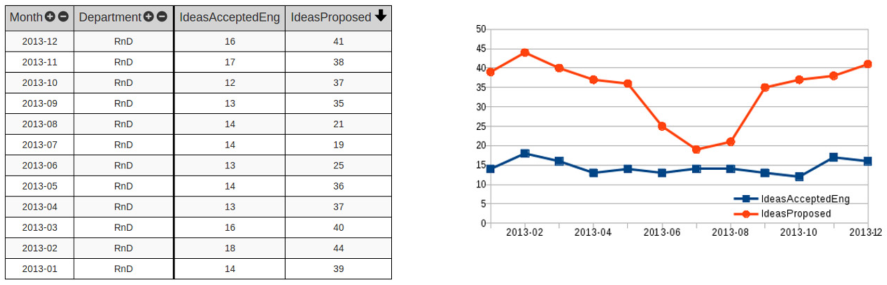
Fig. 2. Interface of the system for extended drill-down

Project [19,20]. As logic programming system we refer to XSB[2], which extends conventional Prolog systems with an operational semantics based on tabling, i.e., a mechanism for storing intermediate results and avoiding to prove sub-goals more than once. XSB also provides interfaces to Java, through which service interfaces are written and calls to Prolog rules are managed, and MDQs are finally translated in SQL. We refer to MySQL to store all the cube elements that are in the enterprise's data mart, without adding pre-aggregations.

Figure 2 shows the interface with the query specification form and the visualization of results. The query is aimed to analyse the monthly trend of $IdeaYield$ in 2013 for the RnD Department: $\langle false, \{IdeaYield\}, \{Month, Department\}, \{\{2013-01, \ldots, 2013-12\}, \{RnD\}\}, \{\}\rangle$. The result is shown both as a table and as a chart. Symbols near to the labels of levels enable classical drill-down/roll-up operators, while the arrow near $IdeaYield$ enables the indicator drill-down. The chart enlightens a peak in July 2013. In order to understand the reason for such a variability, the analyst performs an indicator drill-down on $IdeaYield$. The indicator has been chosen since it allows to demonstrate both rewriting rules and formula inference.

The operator rewrites the query by replacing $IdeaYield$ with its dependencies as in the knowledge base (see Table 1), namely $IdeasAcceptedEng$ and $IdeasProposed$. Since the latter is not in the DM, the equivalence rule is used and the query becomes: $\langle false, \{IdeasAcceptedEng, (NumInternalIdeas + \frac{CrowdInv}{AvgCostsExtIdeas})\}, \{Month, Department\}, \{\{2013-01, \ldots\}, \{RnD\}\}, \{\}\rangle$.

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



Month ⊕⊖	Department ⊕⊖	IdeasAcceptedEng	IdeasProposed ↓
2013-12	RnD	16	41
2013-11	RnD	17	38
2013-10	RnD	12	37
2013-09	RnD	13	35
2013-08	RnD	14	21
2013-07	RnD	14	19
2013-06	RnD	13	25
2013-05	RnD	14	36
2013-04	RnD	13	37
2013-03	RnD	16	40
2013-02	RnD	18	44
2013-01	RnD	14	39

Fig. 3. The result of $IdeaYield$ drill-down

The result is shown in Figure 3. The cause of the trend of $IdeaYield$ is now clear: it is due to the decreasing trend of *IdeasProposed*, while *IdeasAcceptedEng* is almost constant. The analyst can iteratively perform classical OLAP operators and indicator drill-down to refine the result. It is to be noted that, since *IdeasAcceptedEng* is atomic, it is not possible to perform a further indicator drill-down on it; hence the arrow icon is not shown.

In order to evaluate the cost of the novel operator, we observe that the main steps required to perform an indicator drill-down on *ind* are: (1a) searching the formula of *ind* in the knowledge base, or (1b) inferring any valid formula for *ind* (i.e. $\mathcal{F}(ind, C)$) and (2) executing the query over the data mart, until a rewriting succeeds in data retrieval. Steps (1a) and (1b) concern the rewriting of the query, and their costs depend on the number of indicators in the knowledge base, and on the structure of their formulas. The cost of step (2) depends on the cardinality of data, their schema and the adopted management system. It is noteworthy that these parameters do not affect the cost of other steps. Here, we discuss the costs due to query rewriting, which is the cost added by the proposed operator to the classical execution of a query.

Since step (1a) has negligible cost compared to (1b), the complexity of the query rewriting is comparable to that of inferring $\mathcal{F}(ind, C)$ that, in the worst case, corresponds to all possible rewritings of the original formula by traversing all the dependencies' paths. In order to provide an evaluation of these costs we give the average execution time of $\mathcal{F}(ind, C)$ over each *ind* in the knowledge base, for a real and a synthetic scenario[3].

In the real-world scenario, the knowledge base representing the business domains of the BIVEE project is characterized by 356 indicators. The dependency tree has on average 2.67 operands per indicator, height 5 for the root node (i.e. the number of layers of indicators in the tree) and average height 3.14. In this situation, the average execution time of $\mathcal{F}(ind, C)$ is 219ms. We recall that 100ms is about the limit for making the user feel that the system is reacting instantaneously.

[3] Experiments have been carried on an Intel Xeon CPU 3.60GHz with 3.50GB memory, running Windows Server 2003 SP 2.

A synthetic knowledge base has been generated to perform more extensive tests. In particular, we have generated 10 different random trees with height 5, where each indicator is calculated on the basis of 4 random indicators of the lower layer; the number of operands in the formula is fixed. This kind of tree has 1365 different indicators. The average execution time of $\mathcal{F}(ind, C)$ mediated over the 10 trees is 431ms, with 449ms as maximum value. We believe these execution times are perfectly in line with the notion of On Line Analytical Processing, also in the view of possible optimizations of the system.

6 Conclusions

The paper proposed an extension of the data cube model to take into account the structure of an indicator given in terms of a formula. The extension allows to introduce a novel drill-down operator able to increase the detail of a measure of the data cube along the tree of indicators dependencies. Relations between the aggregation function and the formula are taken into consideration, so that the novel and the classic drill-down can be integrated. The logic representation adopted is a powerful way to reason over the tree of indicators' dependencies to calculate measures not explicitly provided in a data mart through formula rewriting. This approach can extend existing DB management systems, as queries can be rewritten either in SQL and executed on relational database, like the prototype shown, or in MDX queries on traditional OLAP systems. The evaluation of a prototype on real and synthetic scenarios enlightens the effectiveness and efficiency of the approach. For the sake of simplicity, the presentation assumed that the data mart schema adopts the knowledge base terminology to define measures, but the model can be simply extended with mapping predicates to relax this assumption. Although only indicators with the analytic expressions managed by PRESS can be represented, this does not limit the model applicability since other kinds of indicators (e.g. qualitative indicators) can be introduced as atomic. We plan to study extensions of the theory towards more complex expressions manipulation. Other extensions regard the representation of relational algebra expressions as indicator's formulas, and of different aggregation functions for different indicator's dimensions.

Acknowledgments. This work has been partly funded by the European Commission through the FoF-ICT Project BIVEE (No. 285746). The authors wish to thank project partners for providing data useful for the case study, and Haotian Zhang for the implementation of the interface.

References

1. Kaplan, R.S., Norton, D.P.: The Balanced Scorecard: Measures that Drive Performance. Harvard Business Review 70, 71–79 (1992)
2. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Information Systems 35, 505–527 (2010)

3. Ackoff, R.L.: Management misinformation systems. Management Science 14 (1967)
4. Lakshmanan, L.V.S., Pei, J., Zhao, Y.: Efficacious data cube exploration by semantic summarization and compression. In: VLDB, pp. 1125–1128 (2003)
5. Neumayr, B., Anderlik, S., Schrefl, M.: Towards Ontology-based OLAP: Datalog-based Reasoning over Multidimensional Ontologies. In: Proc. of the Fifteenth International Workshop on Data Warehousing and OLAP, pp. 41–48 (2012)
6. Niemi, T., Toivonen, S., Niinimäki, M., Nummenmaa, J.: Ontologies with semantic web/grid in data integration for olap. Int. J. Sem. Web Inf. Syst. 3, 25–49 (2007)
7. Huang, S.M., Chou, T.H., Seng, J.L.: Data warehouse enhancement: A semantic cube model approach. Information Sciences 177, 2238–2254 (2007)
8. Priebe, T., Pernul, G.: Ontology-Based Integration of OLAP and Information Retrieval. In: Proc. of DEXA Workshops, pp. 610–614 (2003)
9. Pedrinaci, C., Domingue, J.: Ontology-based metrics computation for business process analysis. In: Proc. of the 4th International Workshop on Semantic Business Process Management, pp. 43–50 (2009)
10. Xie, G., Yang, Y., Liu, S., Qiu, Z., Pan, Y., Zhou, X.: EIAW: Towards a Business-Friendly Data Warehouse Using Semantic Web Technologies. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 857–870. Springer, Heidelberg (2007)
11. 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)
12. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: OLAP Query Reformulation in Peer-to-peer Data Warehousing. Inf. Sys. 37, 393–411 (2012)
13. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Software & Systems Modeling (2012)
14. del-Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Defining process performance indicators: An ontological approach. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 555–572. Springer, Heidelberg (2010)
15. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: Proc. of the ACM 11th International Workshop on Data Warehousing and OLAP, DOLAP 2008, pp. 81–88 (2008)
16. Chesbrough, H.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business Press, Boston (2003)
17. 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, 29–53 (1997)
18. Neumayr, B., Schrefl, M.: Multi-level conceptual modeling and OWL. In: Heuser, C.A., Pernul, G. (eds.) ER 2009. LNCS, vol. 5833, pp. 189–199. Springer, Heidelberg (2009)
19. Diamantini, C., Potena, D., Storti, E.: A logic-based formalization of KPIs for virtual enterprises. In: Franch, X., Soffer, P. (eds.) CAiSE Workshops 2013. LNBIP, vol. 148, pp. 274–285. Springer, Heidelberg (2013)
20. Diamantini, C., Potena, D., Proietti, M., Smith, F., Storti, E., Taglino, F.: A semantic framework for knowledge management in virtual innovation factories. International Journal of Information System Modeling and Design 4, 70–92 (2013)