An Ontology-Based Data Exploration Tool for Key Performance Indicators

Claudia Diamantini, Domenico Potena, Emanuele Storti, and Haotian Zhang

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

Abstract. This paper describes the main functionalities of an ontology-based data explorer for Key Performance Indicators (KPI), aimed to support users in the extraction of KPI values from a shared repository. Data produced by partners of a Virtual Enterprise are semantically annotated through a domain ontology in which KPIs are described together with their mathematical formulas. Based on this model and on reasoning capabilities, the tool provides functionalities for dynamic aggregation of data and computation of KPI values through the formula. In this way, besides the usual drill-down, a novel mode of data exploration is enabled, based on the expansion of a KPI into its components.

1 Introduction

In recent years the capability to innovate and collaborate among different enterprises is more and more considered a crucial skill to react more dynamically to market changes and reduce risks [1]. However, one of the main issues to overcome, especially in the management of temporary networks of enterprises (i.e., Virtual Enterprises, or VE), is the integration of heterogeneous data, and the need to evaluate common Key Performance Indicators (KPI) that are capable to measure performances of the whole VE. In such distributed scenarios, besides heterogeneities due to the use of different terminologies, procedures and processes, each enterprise usually adopts a different set of KPIs to measure their own performances. This introduces a new type of heterogeneity hindering the possibility to monitor comparative performances for the whole VE.

This work is conceived within the European project BIVEE[1], that is targeted at supporting Virtual Enterprises in the achievement of common innovation projects. With the aim to address the above-mentioned issues, the development of the BIVEE platform follows an ontology-based approach for data access, that is frequently used to tackle the problem of integrating and accessing heterogeneous sources, abstracting from how data are maintained. In such a way, a data layer is responsible of storing actual KPI data produced by enterprises, while a

[1] http://bivee.eu

R. Meersman et al. (Eds.): OTM 2014, LNCS 8841, pp. 727–744, 2014.

conceptual model is shared among all the partners and provides a common terminology used to express user requests. Finally, the mapping between the two layers allows to rewrite ontological queries in a language suitable for the data layer.

The conceptual model is here represented by KPIOnto, an ontology devoted to formalise the domain of KPIs and their relations in a logic language. Differing from most of available proposals in the Literature (e.g., [2–5]), in BIVEE KPIs are also formally described through the mathematical formulas needed for their calculation starting from other KPIs. In such a way, different heterogeneities among enterprises can be solved by semantic enrichment of data, or by reasoning over the formulas.

On the top of BIVEE platform, in this work we introduce KPI Explorer, a tool designed to provide users with access to KPI data in an intuitive and informative fashion. Through the reference to KPIOnto, the tool supports users in the graphical composition of a query. Unlike traditional reporting softwares or data exploration tools, the back-end functionalities of KPI Explorer, by exploiting the ontology, are aimed at (1) query rewriting and its execution on the Data Storage, but also to (2) dynamically, transparently and automatically calculate KPI values that are not materialized, but that can be inferred from those available, providing more complete results to users; such advanced functions are enabled by reasoning capabilities that exploit ontological relations among concepts as well as manipulation of mathematical KPI formulas. As for the former, for instance, let us suppose that a KPI is measured by an enterprise on a monthly basis. Given that the concept of "quarter" is described in KPIOnto and every quarter is linked to its corresponding months, it is possible to evaluate the KPI by aggregation of these ones. As regards the latter, let us consider two enterprises willing to compare performance about a certain KPI that, however, is provided just by one of the two; given that formulas are explicitly represented, if the KPI can be obtained for the other enterprise through some formula starting from the KPIs that are currently provided, it will be possible to dynamically compute the final value for both of them. Finally, users can refine the results by applying both classical operators like roll-up/drill-down, and a novel "indicator drill-down" operator, that allows to expand an indicator into its components, allowing the user to gain a better understanding of KPI trends.

This work is structured as follows: Section 2 discusses the most relevant contributions in the Literature, while in Section 3 we introduce those modules of the BIVEE architecture that are central to this work, namely the ontology KPIOnto and the Raw Data Handler. Section 4 introduces the Explorer, discussing both back-end functionalities for query rewriting and reasoning aimed at the evaluation of KPIs and at the manipulation of their formulas. The front-end of the tool is also described and some examples are given. Section 5 presents some experimentations carried on to evaluate the efficacy and efficiency of the proposed approach. Finally, Section 6 provides some final remarks.

2 Related Work

The sharing of data coming from distributed, autonomous and heterogeneous sources arises problems of data inconsistencies and asks for methods to integrate them in a unified view. Given that local schemas are independently developed, they usually have different structure and terminology and several syntactic, structural, and semantic conflicts can occur during schema integration [6].

Several approaches have been proposed in the Literature for schema matching, that is the basic problem in database integration. While real data are produced by the sources, a common approach is to rely on a global schema to obtain an integrated and virtual view, through global-as-view (GAV) and local-as-view (LAV) approaches [7]. Recently, due to the explosive growth of distributed applications, the need of semantic integration is becoming more and more critical. In fact, semantic heterogeneity may persist even if both syntactic and schemas heterogeneities do not occur (e.g., naming the same concept differently). Semantic data integration relies on conceptual and machine-understandable representation of data and their relationships to avoid heterogeneities and allow interoperability. The main methodologies used in the Literature refer to a single ontology approach in which all source schemas are mapped to a central ontology, or a multiple-ontologies approach, with mapping among the local schemas and no shared global view, closely related to GAV and LAV approaches [8]. Ontologies are not only useful as a global conceptualization, but also to support the definition of semantic mappings between the local sources and the global ontology, and to support the user in the formulation of high-level queries over the global schema and its rewriting into local queries. Integration is far more complex when the data models of the sources are multidimensional, as in the case of the present work and in the data warehouse field. Multidimensional models are specifically suited to support data analysis rather than perform on-line transactions, and categorize data as facts with associated quantitative measures, or as a hierarchy of dimensions that describe the facts.

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 [9]. The definition of powerful OLAP operators like drill-down directly comes from this model. Semantic representations of the multidimensional model have been recently proposed [9–12] 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. In particular, support to data cube design is considered in [13], while improvement of OLAP functionalities are presented in [4].

Although the complex nature of indicators is well-known, the compound nature of indicators is far less explored. Proposals in [2, 13–18] include in the representations of indicator properties some notion of formula in order to support the automatic generation of customized data marts, the calculation of indicators [13, 14, 16], or the interoperability of heterogeneous and autonomous data warehouses [2]. 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 [17, 18]. These properties are represented by logical predicates (e.g. *isCalculated* [18], *correlated* [17]) 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 formulas is proposed in [15] 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 [19].

3 Knowledge-Centric BIVEE Platform

The BIVEE project relies on a knowledge-centric approach in which the semantic layer, namely the PIKR [20], represents the foundation supporting advanced functions of the whole environment. It maintains semantic metadata for all the entities of the VE (from actors to products, from innovative ideas to production plans) and offers advanced services. In this Section we briefly discuss the back-end modules of BIVEE infrastructures that are more relevant in the context of this paper. They include KPIOnto, the ontology that is used to represent KPIs and dimensions in the platform, and the Raw Data Handler, where data from enterprises are semantically enriched and stored.

3.1 KPI Ontology

A number of Performance Indicators definitions have been introduced, including glossaries provided by researchers and research groups [21,22], and international and national public bodies (e.g. OECD[2]). While these cannot be regarded as proper models, due to their informal nature, also some reference frameworks for supply chain management were proposed, e.g. the Supply Chain Operations Reference model (SCOR) [23] and Value Reference Model (VRM)[3] that are the two most comprehensive and widely adopted (see also [24] for a more detailed list of performance measurement approaches and KPI description).

Therefore, in order to derive the main properties for the development of KPI-Onto [25], an ontology devoted to formally describe indicators, we referred to VRM with some additional contributions from other glossaries, and to the multidimensional model of data warehouse domain. This choice is justified by three main reasons: (1) the VRM explicitly addresses networked enterprises, that is the reference scenario of this work, (2) the model provides the most complete and detailed description of the properties of indicators, and (3) the BIVEE project itself mainly referred to the VRM model to develop its business innovation reference framework [24].

[2] `http://www.oecd.org/std/leading-indicators/glossaryforoecdcompositeleadingindicators.htm`

[3] `http://www.value-chain.org/en/cms/1960`

Following the VRM model, in KPIOnto KPIs are arranged in a taxonomy according to the enterprise area of intervention or process (e.g. customer, corporate governance, supply chain operation, procurement, human resources, finance and accounting), and are described by a set of properties. Among them, we include the unit of measurement, a textual description, its business object (e.g., Cost, Velocity), and the dimensions along which it can be computed (e.g., Time, Product, Organization) according to the multidimensional model.

For what concerns dimensions, they are arranged in a hierarchy of levels through a *partOf* relation, where each level represents a different way of grouping elements of the same dimension [26]. In this way, the "Month" level is partOf "Quarter", while this is partOf "Semester", that in turn is partOf "Year". Istances of levels are called members (e.g,"2013" is member of "Year"), and are related to other instances through the same partOf relation. Through this property, and its inverse hasPart, it is possible to implement the classical roll-up and drill-down operators for analysis of multidimensional cubes, that allow to navigate among levels to aggregate and disaggregate values.

A KPI can be either atomic or compound, i.e. it can be computed from other lower-level indicators. In this case, dependencies of compound KPIs on its operands are defined by means of algebraic expressions, that is a *Formula* capable to express the semantics of an indicator. A formula describes how a compound indicator is calculated. In KPIOnto each formula is characterized by an aggregation function, the specification of how the formula is presented, the semantics (i.e., the mathematical meaning) of the formula, and references to its components, which are in turn (formulas of) indicators. For instance, $Costs\_DeliverPhase$, that measures all costs related to delivery phase, is compound and has the following formula $Costs\_Ship + Costs\_Pack + Costs\_OrPre + Costs\_Deliv$, while $Costs\_Ship$ is atomic and has no associated formula. In Figure 1 an excerpt of KPIOnto is shown, in which the indicator "Costs_DeliverPhase" is described together with a portion of the Indicator taxonomy. Relations between the indicator and its dimensions are shown, including a fragment of the Dimension hierarchy. As discussed in the experimental Section, the relations among indicators that are encoded in formulas can be represented as a *formula graph*, where each node is an indicator, linked to the components of its formula through edges.

While the descriptive information of the ontology are represented in OWL2[4], KPI formulas are directly represented as algebraic expressions and stored in a repository of formulas that relies on the W3C Recommendation MathML[5] and the emerging standard OpenMath[6]. They are two XML-based standards widely used for rendering equations in web browsers. They provide a language to represent mathematical formulas capable, respectively, to describe the content/presentation of a generic formula and the semantics of mathematical objects. Hence, through such languages, each formula is fully represented together with its dependencies.

[4] `http://www.w3.org/TR/owl2-overview/`
[5] `http://www.w3.org/Math/`
[6] `http://www.openmath.org/`

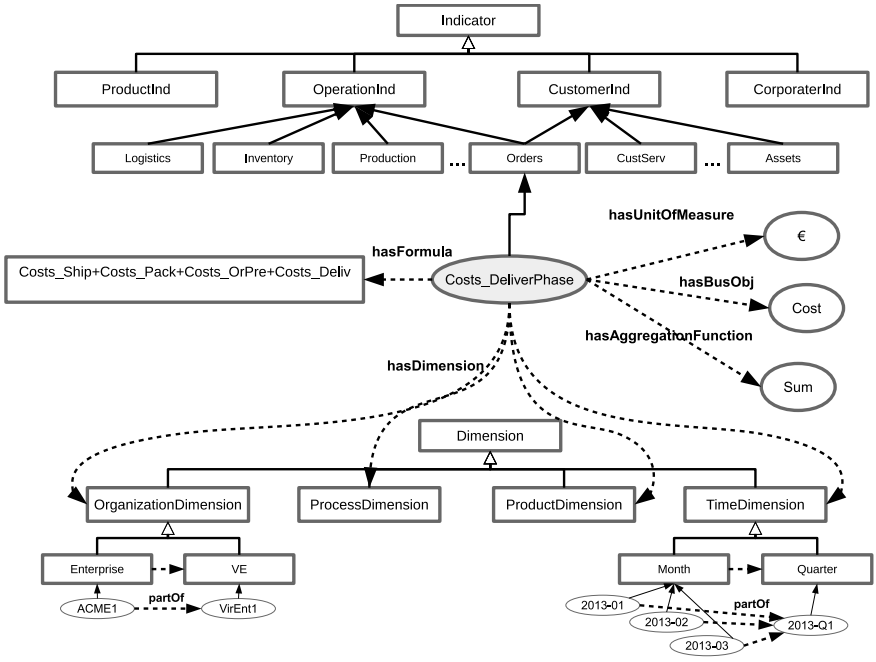

Fig. 1. Excerpt of KPIOnto

The knowledge available in KPIOnto is exploited to support advanced functionalities of the platform, like those described in next Section. In order to work with different typologies of languages and perform reasoning, we refer to Logic Programming as a common logical layer, for its capability to manipulate both OWL2 axioms and mathematical equations. In particular, reasoning functionalities have been implemented in Prolog, while XSB[7] was chosen as logic programming database system for its efficiency. Basic functionalities are mainly aimed to enable manipulation of mathematical formulas, and targeted to achieve the following reasoning tasks:

- formula rewriting, simplification and resolution of equations, adapted from PRESS (PRolog Equation Solving System) [27], a formalization of algebra in Prolog. Such functions are also used to infer, for a given indicator, all the possible rewritings of a formula;
- checking of dependencies among formulas;
- multidimensional query handler, described in Subsection 4.1.

3.2 Raw Data Handler

The service platform and Raw Data Handler (RDH) are aimed to address data gathering and storage, together with the management of data interoperability

[7] `http://xsb.sourceforge.net/`

Table 1. Example of table for Costs_Ship in the Data Storage

VE	*Enterprise*	*Year*	*Semester*	*Quarter*	*Month*	*Week*	*Cat*	*Product*	*Value*
VirEnt1	ACME1	2013	2013-S1	2013-Q1	2013-01	NULL	Armchair	NULL	6.23
VirEnt1	ACME1	2013	2013-S1	2013-Q1	2013-02	NULL	Armchair	Mod_245	1.29
VirEnt1	ACME2	2013	2013-S1	2013-Q1	NULL	NULL	Armchair	Mod_245	2.50
VirEnt1	ACME2	2013	2013-S1	2013-Q2	NULL	NULL	Armchair	NULL	13.22
VirEnt1	ACME2	2013	2013-S1	2013-Q2	NULL	NULL	Table	NULL	14.97

among the heterogeneous enterprises cooperating in a Virtual Enterprise. Data gathering and its semantic leverage are achieved through ETL modules, that extract all the relevant information from distributed data sources, and transform data items by providing the needed semantic enrichment. This last is performed by domain experts that annotate the schema of each source with respect to ontological concepts. It has to be noted that, although the mapping is done in a manual way, it is required only in the setup phase of the VE. Morover, semantic-based tools are available in the ETL module to guide the user in the definition of mappings. Finally, the platform manages data loading into a Data Storage (DS), thus building a materialized centralized database that is updated whenever new data are made available by enterprises. In this work we refer to MySQL as database management system.

Each indicator described in KPIOnto is represented in the DS by a dedicated table, whose schema is defined by the indicator value, the specification of the enterprise and the VE, together with the related dimensions. It has to be noted that, in general, each enterprise provides data about only a subset of all possible indicators, hence resulting in a sparse DS. For what concerns dimensions, the schema contains a column for each level of every compatible dimension. For instance, a fragment of the schema for Costs_Ship is shown in Table 1, in which some example records for two different fictitious enterprises (ACME1, ACME2) are also represented. Each record shows a KPI value (in thousands of euros) that was measured at a different time and for a different product, while NULL values are reported whenever the KPI value is not specified for a certain level because it was pre-aggregated at a higher level (e.g. in the first, Week is NULL since the value refers to overall shipping costs measured in the whole 2013-01 and for the whole category Armchair, while the second refers to a specific product).

It has to be noted that several conflicts may occur while mapping enterprise data to KPI described in KPIOnto. In fact, besides usual conflicts about names or dimensions (e.g. homonymous indicators, or the same indicator measured along different dimensions), also conflicts about measures can occur (e.g., the same indicator calculated through different formulas). For lack of space we do not delve into further details on strategies to reconcile such situations, and refer the interested reader to [28], and to [24] for more general details about RDH.

4 KPI Explorer

In order to access KPI data, a uniform view over them is needed, independently of the original data sources. The KPI Explorer is a tool aimed to allow users to

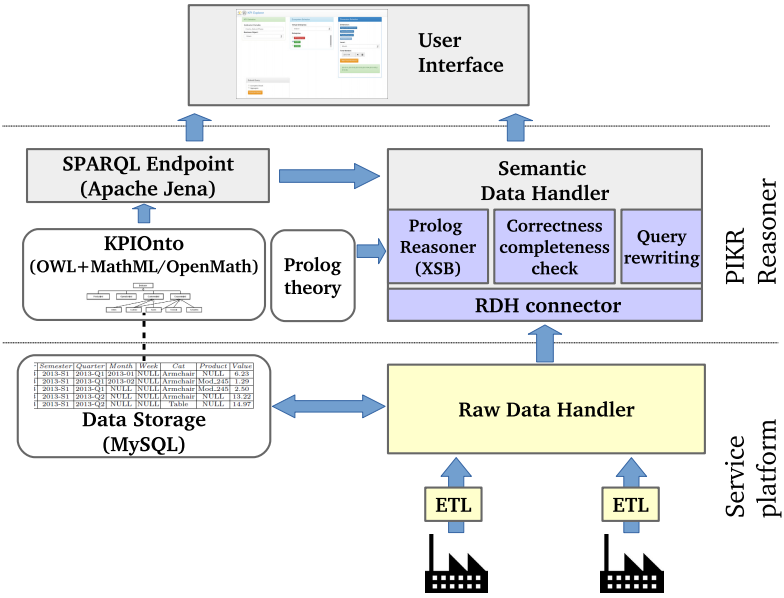

Fig. 2. Architecture of KPI Explorer and back-end services

visually compose a request expressed in terms of the common reference vocabulary for the platform, i.e. the KPIOnto, relying on back-end services that rewrite and resolve such requests as queries over the RDH.

Following the underlying model, a multidimensional query (MDQ) is posed by specifying the following information:

- the name of the requested indicator (e.g., Costs_DeliverPhase),
- for each compatible dimension, the specific level on which to aggregate (e.g., Cat for ProductDimension and Month for TimeDimension),
- an optional set of members on which to filter (e.g., {Armchair} for Cat and {2013-01,...,2013-12} for Month),
- the name of the requested VE and the names of the requested enterprises (e.g., ACME1 and ACME2),
- an optional flag specifying whether to obtain results for each enterprise separately or to aggregate them.

The output of a MDQ is rendered as a table containing all the results that are available in (or that can be dynamically computed from) the Data Storage.

From an architectural point of view, as shown in Figure 2, the KPI Explorer relies on a graphical user interface and on the following components:

a) KPIOnto endpoint, a service relying on Apache Jena[8] to query the ontology through the W3C Recommendation SPARQL language, in order to load from the ontology all the information to fill the fields in the GUI;

[8] `http://jena.apache.org`

b) Semantic Data Handler, the main back-end functionality that manages the execution of a multidimensional query on the Data Storage.

At first, (a) all the needed information about indicators and dimensions are retrieved from the ontology, in order to setup the user interface. After the user has composed a query, this is sent to the Semantic Data Handler (b), that manages its rewriting in SQL and execution over the Data Storage. Once obtained the results, the user can refine them by performing drill-down of either a level or the indicator. While the former is a typical operator in data warehouses, the latter enables a novel mode of exploring KPI data, allowing to re-execute the same query over the dependencies of the original requested KPI, as shown in the example in Subsection 4.2.

The following Subsections discuss in more details the Semantic Data Handler service and the GUI, and provide some examples.

4.1 Semantic Data Handler

The Semantic Data Handler (SDH) is a back-end functionality deployed as a web service within the PIKR, that handles multidimensional queries and embeds reasoning techniques for advanced search of KPI data. Starting from a MDQ expressed with the KPIOnto terminology, the SDH proceeds with its interpretation and checks its correctness, both syntactically (by verifying whether it is well-structured and valid with respect to an XML Schema) and semantically (e.g., by verifying if all terms are actually valid ontological concepts, or if every specified member belongs to the correct level).

Through a query rewriting mechanism, the SDH produces a new query compliant with the specific database management system[9], that is then passed to the RDH for retrieving actual data.

However, in the context of the BIVEE project we do not assume the Data Storage to be complete. In fact, data are usually sparse because each enterprise provides only a subset of all indicators, and for each indicator only few values for every possible combination of members. Furthermore, there is no a priori agreement on the aggregation level for every dimension: e.g., one enterprise can provide data pre-aggregated on a monthly basis, while another on a quarterly basis, preventing the comparison of their performances.

To solve this kind of heterogeneity, an advanced feature of the SDH, namely *completeness check*, is devised to support the dynamic computation of those values that are not materialized, by exploiting the semantic description of the dimensions and the KPIs provided by KPIOnto. Such a functionality, for each result item required by the user (i.e. for every combination of members in the request) that is not available in the Data Storage, recursively applies a set of rewriting rules based on Logic Programming. To this purpose, the content of KPIOnto (indicator/dimension details and formulas) is translated as Prolog facts. Such rules allow to automatically (1) calculate an indicator from its formula (if the

[9] Currently the RDH relies on SQL and MySQL, but the approach is valid for any alternative solution, like MDX queries on OLAP systems.

formula is not defined, the Prolog reasoner tries to infer it by manipulating and reverting already existing formulas), or (2) aggregate the value for a member of a level from its lower-level members.

To give an example of the two rules, let us consider the following MDQ:

$\langle$Costs_DeliverPhase,{Cat,Quarter},{{Armchair},{2013-Q1}},{ACME$_x$,ACME$_y$}$\rangle$.

If for ACME$_y$ there is no value for the Cat Armchair and for the Quarter 2013-Q1, the system tries to apply the rewriting rules. Through rule (1) by exploiting the formula $Costs\_DeliverPhase = Costs\_Ship + Costs\_Pack + Costs\_OrPre + Costs\_Deliv$, a new query is automatically generated for each dependency of the KPI. In this case the new queries are:

$\langle$ Costs_Ship, {Cat,Quarter}, {{Armchair},{2013-Q1}},{ACME$_y$} $\rangle$
$\langle$ Costs_Pack, {Cat,Quarter}, {{Armchair},{2013-Q1}},{ACME$_y$} $\rangle$
$\langle$ Costs_OrPre, {Cat,Quarter}, {{Armchair},{2013-Q1}},{ACME$_y$} $\rangle$
$\langle$ Costs_Deliv, {Cat,Quarter}, {{Armchair},{2013-Q1}},{ACME$_y$} $\rangle$

If such queries are satisfiable, then the four values are retrieved and aggregated by using the formula. The result is the value for $Costs\_DeliverPhase$ requested by the original query.

An alternative solution, according to rule (2), is to drill-down member 2013-Q1 or member Armchair to their lower members[10]. This implies, for every member, to create a new query and (if and once solved) to compute the required value from those available by means of the aggregation function of the indicator (if any). In the example, if the system drills-down 2013-Q1 to its components 2013-01, 2013-02 and 2013-03, and if the aggregator for the KPI is "Sum", then it is possible to calculate the value as a summation of the results of the following queries:

$\langle$ Costs_DeliverPhase, {Cat,Month}, {{Armchair},{2013-01}},{ACME$_y$} $\rangle$
$\langle$ Costs_DeliverPhase, {Cat,Month}, {{Armchair},{2013-02}},{ACME$_y$} $\rangle$
$\langle$ Costs_DeliverPhase, {Cat,Month}, {{Armchair},{2013-03}},{ACME$_y$} $\rangle$

The result set obtained with the completeness check is always a superset of what can be generated without. However, due to its computational complexity, it is noteworthy that the running time of the completeness check procedure is related to the size of KPIOnto and the Data Storage. We refer the interested reader to [29] for a detailed discussion on the indicator drill-down rule.

4.2 User Interface

The front-end of the KPI Explorer enables users to graphically compose a MDQ with the aim to analyse KPIs for one or more enterprises. To present the main

[10] Relations among members that are used in this case are the partOf introduced in Section 3.1.

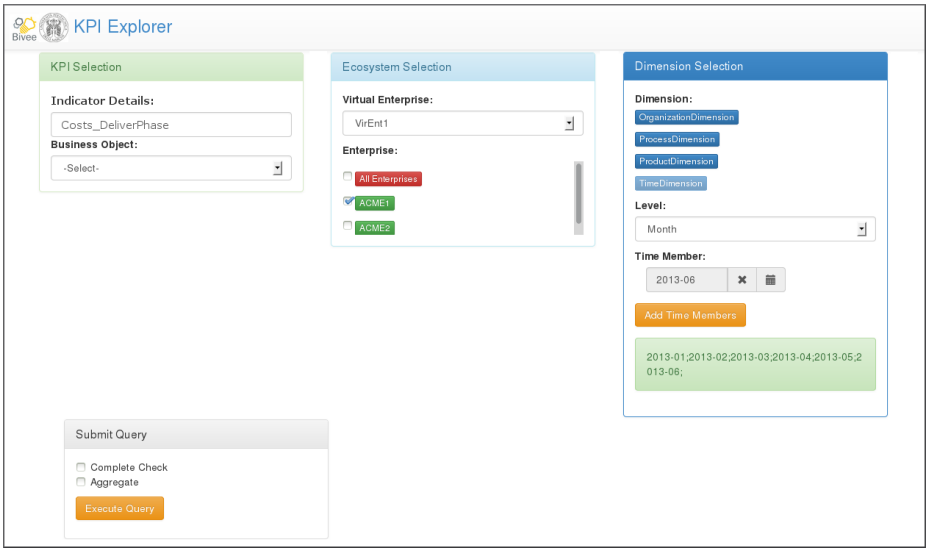

Fig. 3. KPI Explorer: composition of a query

Cat	Month	Enterprise	Costs_DeliverPhase
Armchair	2013-01	ACME1	15.7
Armchair	2013-02	ACME1	15.1
Armchair	2013-03	ACME1	15.3
Armchair	2013-04	ACME1	14.8
Armchair	2013-05	ACME1	13.9
Armchair	2013-06	ACME1	14.3

Fig. 4. KPI Explorer: result table

features, let us consider the following case study, in which a user wants to analyse the *Costs_DeliverPhase* for enterprise ACME1 belonging to the Virtual Enterprise $VirEnt1$.

As she specifies the required indicator within the search-box (see Figure 3), the rest of the form is dynamically loaded, including the list of compatible dimensions. In the example, the user decides to analyse the KPI with respect to *ProductDimension* and *TimeDimension*. For what concerns the former, she selects *Cat* level and specifies *Armchair* as member on which to filter the results. As for the latter, she specifies the first six months in 2013. Unselected dimensions are not part of the MDQ. After the execution of the query by the SDH, results are represented in a table, as shown in Figure 4.

The user notices a drop in the costs during spring 2013. In order to understand the reason behind this variability, she can refine the analysis by performing a drill-down of either a level or the KPI, by clicking on the "plus" symbol next to each label. As for the former (e.g., Month), the GUI retrieves from KPIOnto the lower level(s) (Week in this case), and performs a query at that

Cat	Month	Enterprise	Costs_DeliverPhase	Costs_Ship	Costs_Pack	Costs_OrPre	Costs_Deliv
Armchair	2013-01	ACME1	15.7	6.23	2.15	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	3.09	3.95
Armchair	2013-04	ACME1	14.8				
Armchair	2013-05	ACME1	13.9	5.73	2.17	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	3.13	

Fig. 5. KPI Explorer: drill-down of Costs_DeliverPhase

level. As regards the latter, the GUI calls the KPIOnto endpoint to retrieve the formula for *Costs_DeliverPhase*; then, for each of its components, a new query is created and sent to the SDH. Results about this last case, as shown in Figure 5, contribute to improve the user awareness about which specific sub-indicator mostly affected the final cost. To further deepen the analysis, the user performs a new drill-down on the sub-indicator Costs_Pack, as shown in Figure 6(a). Empty cells in the results stand for unavailable data, that have not been provided by the enterprise. For such a reason, the user executes again the last query enabling the "completeness check" option. As shown in Figure 6(b), for some of the previously empty cells now a value is shown (red boxes), inferred by using the rewriting rule (1) introduced in Section 4.1. In detail, to retrieve the value for *Materials_Pack* referring to 2013-01, the reasoner exploits the formula $Costs\_Pack = Materials\_Pack + Others\_Pack + LabourCosts\_Pack$ and solves it for *Materials_Pack*. To make an other example, the value for *Others_Pack* referring to 2013-05 is obtained as follows: at first the equation *Costs_DeliverPhase* was solved for *Costs_Pack*; then, the reasoner exploits the formula for *Costs_Pack* shown in the previous example.

As a second example, the user wants to compare performances about the same KPI for ACME1 and ACME2 for the first two quarters of 2013 and for Armchair category. However, the two enterprises provided different data with different granularity. In fact, ACME1 provided measures of both *Costs_DeliverPhase* and all its components *Costs_Ship*, *Costs_Pack*, *Costs_OrPre* and *Costs_Deliv*. On the other hand, ACME2 provided values for all of them, except for the *Costs_DeliverPhase*. Moreover, ACME1 measures values on a monthly basis, while ACME2 on a quarterly basis. Since no direct comparison can be made due to such heterogeneities, the user enables the "completeness check". In such a way, the SDH exploits the Prolog-based rewriting rules discussed above. In more details, rule (1) is applied to calculate *Costs_DeliverPhase* for ACME2 starting from its dependencies, while rule (2) is exploited to aggregate months in quarters for ACME1. As a result, the user obtains the required KPI at the given levels, as shown in Figure 7.

(a)

Cat	Month	Enterprise	Costs_DeliverPhase	Costs_Ship	Costs_Pack	Materials_Pack	Others_Pack	LabourCosts_Pack	Costs_OrPre	Costs_Deliv
Armchair	2013-01	ACME1	15.7	6.23	2.15		0.1	1.31	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		0.69		1.23	3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	0.62	0.63	0.98	3.09	3.95
Armchair	2013-04	ACME1	14.8			0.68	0.33	1.14		
Armchair	2013-05	ACME1	13.9	5.73	2.17	0.71		1.2	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	0.72	0.4	1.07	3.13	

(b)

Cat	Month	Enterprise	Costs_DeliverPhase	Costs_Ship	Costs_Pack	Materials_Pack	Others_Pack	LabourCosts_Pack	Costs_OrPre	Costs_Deliv
Armchair	2013-01	ACME1	15.7	6.23	2.15	0.74	0.1	1.31	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		0.69	0.11	1.23	3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	0.62	0.63	0.98	3.09	3.95
Armchair	2013-04	ACME1	14.8			0.68	0.33	1.14		
Armchair	2013-05	ACME1	13.9	5.73	2.17	0.71	0.26	1.2	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	0.72	0.4	1.07	3.13	

Fig. 6. KPI Explorer: drill-down of Costs_Pack (a) without and (b) with completeness check

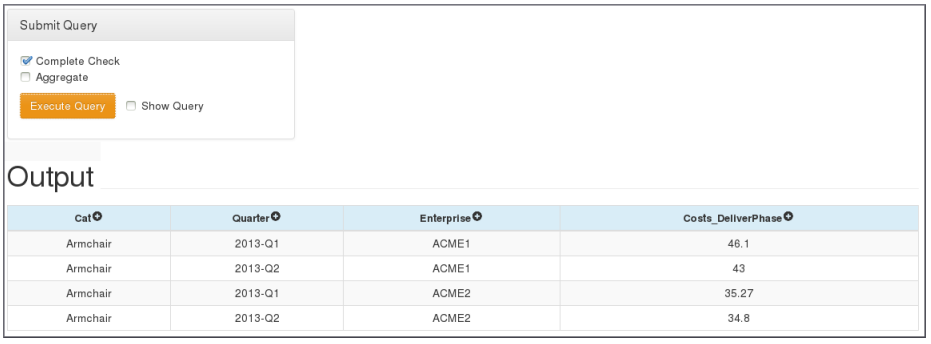

Fig. 7. KPI Explorer: results for ACME1 and ACME2 with completeness check

5 Evaluation

The goal of this Section is to provide an evaluation of costs and benefits of the proposed approach, respectively in terms of efficacy and response time.

As for the former, we evaluated how larger it is the number of indicators that can be dynamically computed through the reasoning mechanism, with respect to those directly available in the Data Storage. To evaluate how the result depends on the size of the ontology, the experimentation was performed on a set of synthetic ontologies with an increasing number of indicators, and then on the real-world ontology used in the BIVEE project. Synthetic ontologies were au-

tomatically generated by varying two parameters of the corresponding formula graph: the number of operands per indicator (*op*), and the maximum number of levels (*lev*). For sake of simplicity, all formulas for the indicators are defined as summation of other randomly chosen indicators. For instance, a 1-level ontology contains only one top indicator that is computed by summing *op* indicators; a 2-levels ontology is formed by one top level indicator, which is computed on *op* indicators in the middle layer that, in turn, are computed on *op* low-level indicators. The number of indicators in each ontology is given by $\sum_{i=0}^{lev} op^i = \frac{1 - op^{lev+1}}{1 - op}$, ensuring that each indicator has exactly *op* number of operands. Table 5 reports the list of generated ontologies with the number of indicators.

Table 2. Size of the generated ontologies: operands, levels and number of indicators

op	2	3	4	5	2	3	2	2
lev	2	2	2	2	3	3	4	5
#indicators	7	13	21	31	15	40	31	63

Let F_i be the set of all possible inferable formulas for the indicator i of the given ontology O; the procedure followed in the experimentation is:

For $k = 1$ to 100
1. randomly define a subset S of indicators in O
2. for each indicator $i \in O$
   check whether i is computable over S

An indicator i is *computable* over the set S if at least a formula $f \in F_i$ exists such that all operand of f are indicators in S, hence f can be computed by using only available indicators. Given that results depend on the number of indicators actually materialized, for each ontology the procedure is executed multiple times, namely 100: each time a randomly selected subset of indicators in the ontology is assumed to be actually available in the Data Storage.

In order to keep the complexity of the experimental procedure under control, for each indicator, the maximum number of formulas the reasoner inferred was set to 50. Figure 8 shows the rate of computable indicators (i.e. the ratio between the number of computable indicators and the number of indicators in the ontology) vs. the rate of available indicators in Data Storage (i.e. the ratio between the number of available indicators in the Data Storage and the number of indicators in the ontology). For each ontology we drew a different line in the figure. The bisector (dotted line) represents the baseline, where the number of computable indicators is equal to number of those available. In this case only indicators that are actually available in the Data Storage can be computed, i.e. points on the bisector represent the situation where no reasoning is performed. We'd like to note that all results are above the bisector, demonstrating the efficacy of the approach. When few indicators are available (i.e. the rate of available

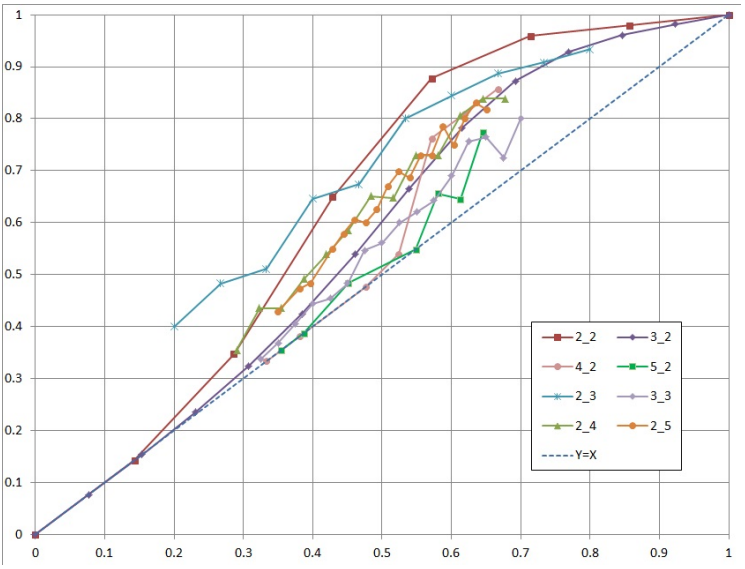

Fig. 8. Rate of computable indicators vs. rate of available indicators in Data Storage

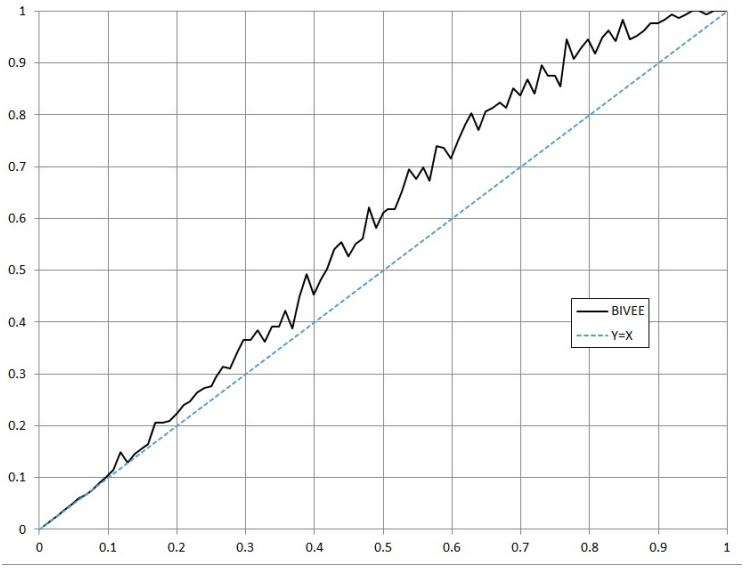

Fig. 9. Rate of computable indicators vs. rate of available indicators in Data Storage, for KPIOnto

indicators is less than 0.15), the approach does not outperform the baseline, hence the reasoning mechanism is ineffective. On the other side, with a rate of available indicators higher than 0.65, we are able to compute more than the 80% of indicators in the ontology. The same remarks can be made also for results obtained over KPIOnto, the real-world ontology used in the BIVEE project, as shown in Figure 9. This ontology takes into account 356 KPIs and its formula graph is formed by a set of disconnected graphs, with different levels (from 1 to 5) and operands (from 2 to 4). On average, the graph has 3.14 levels and 2.67 operands per indicator.

Table 3 shows the average execution time of the reasoning mechanism for inferring all formulas for an indicator. Experiments were carried on an Intel i7 CPU 2.20GHz with 8GB memory, running Windows 7 64bit. For some of the ontologies (e.g., lev=3 and op=4), it was not feasible to conduct the tests. We'd like to note that the execution time does not increase with the number of indicators, but it depends on the topology of the ontology, i.e. the way in which the indicators are connected to each other. On average, the inference of all formulas required less than 1.3s. As regards KPIOnto, the execution time is on average around 6s. However, it is notewhorty that real scenarios do not usually involve extracting the complete set of formulas for an indicator, reducing in this way the overall execution time.

Table 3. Average execution times for inferring all formulas for an indicator (ms)

	Lev			
Op	2	3	4	5
2	69.43	211.69	651.87	2083.38
3	528.93	1699.63	3792.43	n/a
4	1294.05	n/a	n/a	n/a
Average	*1291.43*			
KPIOnto	*5968.14*			

6 Conclusion

In this paper, we presented a tool for exploration of data related to Key Performance Indicators, provided by multiple enterprises collaborating in a shared innovation project, in the context of a Virtual Enterprise. The tool relies on the architecture of the BIVEE platform, and in particular on a semantic middleware that includes both a domain ontology and a repository where heterogeneous data are annotated and stored. Back-end functionalities of the tool ensure the rewriting of the user query, expressed in ontological terms, and its execution on the Data Storage. Advanced functions enable automatic, dynamic and transparent aggregation of KPIs at any level, and computation of KPI values through their mathematical formulas, by exploiting information in the ontology. As for the front-end, such an approach allows for a novel mode in data exploration, since

users can expand a KPI into its more basic components, thus enabling the possibility to deepen the analysis by browsing the KPI hierarchy, in an effective and efficient way, as shown by the encouraging outcome of the experimentation. Future work will focus on providing more detailed and formal descriptions of the underlying reasoning functionalities, and on performing more comprehensive tests to evaluate the applicability of the proposed approach in real-life scenarios.

Acknowledgments. This work has been partly funded by the European Commission through the FoF-ICT Project BIVEE (No. 285746). The authors wish to thank the Commission for its support and all the project partners for their contribution in the development of various ideas and concepts presented in this paper.

References

1. Chesbrough, H.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business Press, Boston (2003)
2. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: Olap query reformulation in peer-to-peer data warehousing. Information Systems 37, 393–411 (2012)
3. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distrib. Parallel Databases 23, 69–97 (2008)
4. Priebe, T., Pernul, G.: Ontology-Based Integration of OLAP and Information Retrieval. In: Proc. of DEXA Workshops, pp. 610–614 (2003)
5. del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. Information Systems 38, 470–490 (2013)
6. Lee, C., Chen, C.J., Lu, H.: An aspect of query optimization in multidatabase systems. SIGMOD Rec. 24, 28–33 (1995)
7. 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 (2002)
8. Cruz, I.F., Xiao, H.: The role of ontologies in data integration. Journal of Engineering Intelligent Systems 13, 245–252 (2005)
9. Lakshmanan, L.V.S., Pei, J., Zhao, Y.: Efficacious data cube exploration by semantic summarization and compression. In: VLDB, pp. 1125–1128 (2003)
10. 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)
11. 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)
12. Huang, S.M., Chou, T.H., Seng, J.L.: Data warehouse enhancement: A semantic cube model approach. Information Sciences 177, 2238–2254 (2007)
13. 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.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 857–870. Springer, Heidelberg (2007)

14. 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)
15. Kehlenbeck, M., Breitner, M.: 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)
16. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Software & Systems Modeling (2012)
17. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Information Systems 35, 505–527 (2010)
18. 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)
19. 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)
20. Diamantini, C., Potena, D., Proietti, M., Smith, F., Storti, E., Taglino, F.: A semantic framework for knowledge management in virtual innovation factories. Int. J. Inf. Syst. Model. Des. 4, 70–92 (2013)
21. Roubioni, N.: Hypertext glossary of business cycle indicators, `http://people.stern.nyu.edu/nroubini/bci/bci.html`
22. Crosta, F.: Indicatori di performance aziendali: come definire gli obiettivi e misurare i risultati (in italian). Franco Angeli (2005)
23. Supply Chain Council: Supply chain operations reference model. Supply Chain Council (2008)
24. Isaja, M., Diamantini, C., Potena, D., Storti, E., Taglino, F., Smith, F.: BIVEE integrated semantic meta-space: advanced interoperability services. Deliverable 5.3. Bivee European project (2013)
25. 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)
26. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edn. John Wiley & Sons, Inc., New York (2002)
27. Sterling, L., Bundy, A., Byrd, L., O'Keefe, R., Silver, B.: Solving symbolic equations with press. J. Symb. Comput. 7, 71–84 (1989)
28. Diamantini, C., Potena, D., Storti, E.: Data mart reconciliation in virtual innovation factories. In: Iliadis, L., Papazoglou, M., Pohl, K. (eds.) CAiSE 2014 Workshops. LNBIP, vol. 178, pp. 274–285. Springer, Heidelberg (2014)
29. Diamantini, C., Potena, D., Storti, E.: Extending drill-down through semantic reasoning on indicator formulas. In: Bellatreche, L., Mohania, M. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 57–68. Springer, Heidelberg (2014)