Designing Multidimensional Cubes from Warehoused Data and Linked Open Data

Franck Ravat, Jiefu Song, Olivier Teste

Designing Multidimensional Cubes from Warehoused Data and Linked Open Data

Franck Ravat
IRIT-Université Toulouse I Capitole
UMR5505, CNRS
Toulouse, France
ravat@irit.fr

Jiefu Song
IRIT-Université Toulouse I Capitole
UMR5505, CNRS
Toulouse, France
song@irit.fr

Olivier Teste
IRIT-Université Toulouse II Jean Jaurès
UMR5505, CNRS
Toulouse, France
teste@irit.fr

***Abstract*— A Data Warehouse (DW) is widely used as a consistent and integrated data repository in Business Intelligence systems. Under today's dynamic and competitive business context, warehoused data alone no longer provide enough information for decision-making processes. Business analyses should be enhanced by including Linked Open Data (LOD) to offer multiple perspectives to decision-makers. This paper provides a new multidimensional model, named *Unified Cube*, which offers a generic representation for both warehoused data and LOD at the conceptual level. A two-stage process is proposed to build a *Unified Cube* according to decision-makers' needs. As a first step, schemas published with specific modeling languages are transformed into a common conceptual representation. The second step is to associate together related data to form a *Unified Cube* containing all useful information about an analysis subject. A high-level declarative language is provided to enable non-expert users to define the relevance between data according to their analysis needs. To demonstrate the feasibility of the proposed concepts, we show how analyses over data from different sources can be carried out through a *Unified Cube*.**

Keywords—conceptual multidimensional modeling; linked open data; unified business analyses

I. INTRODUCTION

For over a decade, Business Intelligence systems have been widely used to provide complex information during a decision-making process. Operational data are periodically extracted, transformed and loaded in a consistent and integrated repository, called *Data Warehouse* (DW), which organizes data into multidimensional cubes. Decision-making based only on warehoused data gives a partial view over the activities of the organization. Under today's highly competitive business context, additional information coming from the outside of an organization, mostly found on the Web, should also be included in analyses to provide multiple perspectives to decision-makers [1].

Extending business analyses with external data requires knowing the exact semantics of information. Using semantic web formats, *Linked Open Data* (LOD)[1] is designed to publish semantically interconnected and machine-readable data on the Web. By simply accessing data providers, numerous multidimensional LOD can be extracted and used in a decision-making context [2]. However, relevant warehoused data and LOD are scattered in different schemas. During an analysis, decision-makers should navigate among several schemas to gather all useful information. The dispersion of warehoused data and LOD leads to repetitive searches for relevant information among various data sources, which reduces the efficiency of analysis. Furthermore, warehoused data and LOD follow specific models in each domain. The differences between DW and LOD models make it hard for decision-makers to carry out analyses over the two types of data in a unified way, which complicates the analysis tasks.

Facing these issues, our aim is to make full use of all relevant data in a decision-making context. To this end, we represent warehoused data and multidimensional LOD in a unified way and independently of the specific modeling languages of each domain. To facilitate the analysis tasks of decision-makers, the unified representation should only include concepts close to business terms.

In this paper, we describe a generic modeling solution for both warehoused data and LOD. We also define a process of unifying all useful data for analyses. First, we define a conceptual multidimensional model, named *Unified Cube*, which corresponds to a generic representation of data from several sources relating to one analysis subject. Second, based on the generic modeling language of *Unified Cube*s, we propose a two-stage process to (a) transform various schemas into a generic conceptual representation and (b) link together relevant data to form a unified schema for all useful data. At the end of the process, decision-makers obtain a *Unified Cube* containing as much related information as possible to make effective and well-informed decisions. To validate the feasibility of the two propositions, we develop a prototype containing a *Unified Cube* built upon a relational DW and two online LOD datasets. By illustrating how a *Unified Cube* can be queried, we demonstrate the feasibility of analyzing both warehoused data and LOD in a unified way.

The rest of the paper is organized as follows. Section 2 introduces a running example of unified analyses over data from multiple data sources. Section 3 presents the multidimensional modeling language composed of conceptual definitions and related graphical notations for *Unified Cube*s. Section 4 describes the process of building a *Unified Cube* from different schemas in the DW and LOD domains. Section 5 presents some experimental assessments to demonstrate the feasibility of the proposed concepts. Section 6

[1] http://linkeddata.org

discusses related work on unifying relevant element from different schemas in the fields of DW and LOD.

II. Running Example

In a company specializing in air conditioning devices, a decision-maker refers to an internal R-OLAP DW to assess the performance of sales staff. The DW relates to an analysis subject (i.e. fact), named *Sales*, which contains a set of numeric indicators (i.e. measure), namely *quantity*, *unit price* and *revenue* of sales. Each measure can be computed according to three analysis axes (i.e. dimensions): *salesman*, *product* and *date*. In Fig. 1, dimensions are implemented through a set of dimension tables. For instance, the dimension named *product* is composed of four dimension tables, namely *PRODUCT*, *BRAND*, *RANGE* and *SECTOR*. Each dimension table represents a granularity level. The dimension tables at lower levels contain one or several foreign keys pointing to the higher levels. For instance, the dimension table *PRODUCT* reveals the granularity level *P_Key* with two foreign keys leading to its parent levels *BRAND* and *RANGE*. The fact is implemented with a fact table. The set of foreign keys of a fact table points to the lowest granularity levels of the associated dimensions. For instance, the fact table *SALES* includes three foreign keys associated with the lowest granularity level (i.e. *P_KEY*, *S_KEY* and *D_KEY*) of the three related dimensions.

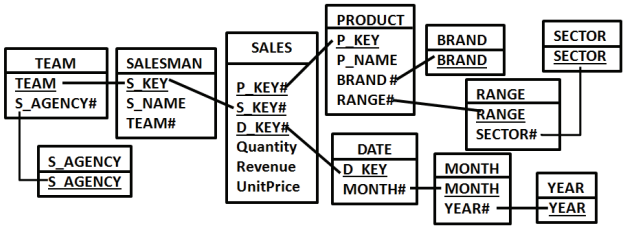


Fig. 1. R-OLAP implementation of the warehoused data.

The R-OLAP DW alone does not provide enough information to support effective and well-informed decisions. The decision-maker must search for additional information to obtain other complementary perspectives over the sales activities. Since she/he notices that the promotion of air-conditioning device is more efficient when the customers in a city experience unusual weather changes. To get more insight into the relationships between the promotion activities and the climatic changes, the decision-maker browses in an online LOD dataset revealing the *monthly* average temperature according to *cities* and *countries* during the promotion period of the company. The LOD are published in RDF *Data Cube Vocabulary* (QB)[2] format, which is the current W3C standard to publish multidimensional statistical data. In brief, a QB *dataset* includes a set of `qb:observations`[3] (close to the definition of *facts* in DW terminology) that describes `qb:measureProperty` (i.e. measures) with related `qb:dimensionProperty` (i.e. dimensions). For instance, in Fig. 2 the `qb:dataset` named *Climate Changes* includes two instances of `qb:observation` identified by `eg:ob1` and `eg:ob2` (cf. lines 15-24). Each instance of `qb:observation` relates an instance of `eg:M_TEMPERATURE` with an instance of associated *dimension*s, namely `eg:GEOGRAPHY` and `eg:TIME`.

```
1  eg:CLIMATE
2      a qb:dataSet ;
3      rdfs:label "Climate Changes"@en;
4      qb:structure :CLIMCHANGE.
5  eg:CLIMCHANGE a qb:DataStructureDefinition;
6      rdfs:label "Data Structure"@en;
7      qb:component
8          [ qb:dimension eg:TIME],
9          [ qb:dimension eg:GEOGRAPHY];
10     qb:component
11         [qb:measureProperty eg:M_TEMPERATURE].
12 eg:M_TEMPERATURE a qb:measureProperty;
13     rdfs:label "Temperature"@en;
14     rdfs:range xsd:decimal.
15 eg:ob1  a qb:Observation;
16    qb:dataSet eg:CLIMATE;
17    eg:TIME eg:ym2015-11;
18    eg:GEOGRAPHY eg:c1;
19    eg:M_TEMPERATURE 12.
20 eg:ob2  a qb:Observation;
21    qb:dataSet eg:CLIMATE;
22    eg:TIME :ym2015-11;
23    eg:GEOGRAPHY eg:co1;
24    eg:M_TEMPERATURE 10.
```

Fig. 2. Extract of the LOD published in QB format.

It is worth noticing that unlike the hierarchical definition of dimension in DWs, a dimension in QB is a non-hierarchical concept. The unique granularity level within a QB dimension is represented by `skos:hasTopConcept` at the schema level. For instance, in Fig. 3 the attributes `eg:CITY` and `eg:COUNTRY` share an unspecified granularity level on the dimension `eg:GEOGRAPHY` (cf. line 29).

```
25 eg:GEOGRAPHY a qb:dimensionProperty;
26     rdfs:label "GEOGRAPHY"@en;
27     qb:codeList eg:attGeo.
28 eg:ATTSGEO a skos:ConceptScheme;
29     skos:hasTopConcept eg:CITY, eg:COUNTRY.
30 eg:CITY a qb:levelProperty;
31     rdfs:label "CITY"@en.
32 eg:COUNTRY a qb:levelProperty;
33     rdfs:label "COUNTRY"@en.
34 eg:c1 a eg:CITY;
35   rdfs:label "TOULOUSE"@en;
36   skos:broader eg:co1.
37 eg:co1 a eg:COUNTRY;
38   rdfs:label "FRANCE"@en.
```

Fig. 3. Dimension *GEOGRAPHY* in the QB datset.

Since the retail sales may compete with the company's promotions in the same catchment area, the decision-maker consults another online LOD dataset about the outlet prices offered by rival retailers, This dataset is published in QB4OLAP; it involves the *retail price* for a type of *merchandise* offered by a *retailer*. In the field of LOD, QB4OLAP is the only vocabulary that covers the most used characteristics of multidimensional models, like multiple granularity levels (i.e. `qb4o:levelInHierarchy`) within multiple aggregation paths (i.e. `qb4o:hierarchyProperty`) and the specification of the aggregation functions associated to a measure (i.e. `qb4o:aggregateFunction`). Even though QB4OLAP covers most features of multidimensional models, the decision-maker has difficulties to explore directly such a schema due to the complex syntax (cf. Fig. 4).

[2] http://www.w3.org/TR/vocab-data-cube

[3] Details about the prefixes are available on http://prefix.cc/

```
eg:RETAIL
    a qb:dataSet ;
    rdfs:label "Outlet"@en ;
    qb:structure eg:RetPrice.
eg:RetPrice a qb:DataStructureDefinition;
    rdfs:label "Retail Price Dataset Structure"@en;
    qb:component
        [ qb4o:level eg:SHOP; qb4o:cardinality qb4o:ManyToOne],
        [ qb4o:level eg:TYPE; qb4o:cardinality qb4o:ManyToOne];
    qb:component
        [qb:measure eg:M_RETAILPRICE; qb4o:aggregateFunction
            qb4o:sum, qb4o:avg, qb4o:min, qb4o:max ].
eg:M_RETAILPRICE a qb:measureProperty;
    rdfs:label "RetailPrice"@en;
    rdfs:range xsd:decimal.
eg:RETAILER a qb:dimensionProperty;
    rdfs:label "RETAILER"@en;
    qb4o:hasHierarchy eg:H_COM, eg:H_CA.
eg:H_COM a qb4o:hierarchyProperty;
    rdfs:label "COMPANY HIERARCHY"@en;
    qb4o:inDimension eg:RETAILER;
    qb4o:hasLevel eg:l_SHOP, eg:l_COMPANY.
eg:H_COM a qb4o:hierarchyProperty;
    rdfs:label "CATCHMENTAREA HIERARCHY"@en;
    qb4o:inDimension eg:RETAILER;
    qb4o:hasLevel eg:l_SHOP, eg:l_CATCHMENTAREA.
eg:l_SHOP a qb4o:levelInHierarchy ;
    qb4o:levelComponent eg:SHOP;
    qb4o:hierarchyComponent eg:H_COM, eg:H_CA.
eg:SHOP a qb:levelProperty;
    rdfs:label "SHOP"@en.
eg:l_CATCHMENTAREA a qb4o:levelInHierarchy ;
    qb4o:levelComponent eg:CATCHMENTAREA;
    qb4o:hierarchyComponent eg:H_CA.
eg:l_COMPANY a qb4o:levelInHierarchy ;
    qb4o:levelComponent eg:COMPANY;
    qb4o:hierarchyComponent eg:H_COM.
eg:Step_l_sTol_ca a qb4o:hierarchyStep;
    qb4o:childLevel eg:l_SHOP;
    qb4o:parentLevel eg:l_CATCHMENTAREA;
    qb4o:cardinality qb4o:oneToMany.
eg:Step_l_caTol_co a qb4o:hierarchyStep;
    qb4o:childLevel eg:l_SHOP;
    qb4o:parentLevel eg:l_CATCHMENTAREA;
    qb4o:cardinality qb4o:oneToMany.
```

Fig. 4. Extract of the LOD published in QB4OLAP format.

Despite sharing some common features, warehoused data and multidimensional LOD follow different models defined by specific modeling languages in each domain at the logical level [3], [4]. The complex syntax of these modeling languages complicates the tasks of analysis, especially for non-expert users. Moreover, since each of schemas provides a partial view over the analysis subject, the decision-maker should look into several schemas one after another to obtain multiple perspectives. Confronted with these issues, the decision-maker wants to build a conceptual multidimensional schema that includes both the data from the R-OLAP DW and the relevant QB and QB4OLAP datasets.

In the remainder of this paper, we will present how a *Unified Cube* can help make full use of both warehoused data and LOD in a decision-making context.

III. Conceptual Modeling of Unified Cubes

In this section, we describe a multidimensional modeling language composed of a set of conceptual definitions of *Unified Cube*s. The modeling language is generic enough to include both warehoused data and multidimensional LOD within a single schema.

Besides the differences in the syntax of modeling languages, the communities of DW and LOD do not share the same principles while publishing data: the domain of DW mainly focuses on the structure of data (i.e. schema), while the LOD domain encourages interconnecting independent instances without necessarily being associated to a data schema. In order to represent both warehoused data and LOD, the multidimensional modeling language should include conceptual definitions for components at both schema level and instance level. Graphical notations are also proposed to facilitate the understanding of non-expert users.

A. Dimension Schema

A *dimension schema represents the structure of an analysis axis* composed *of* attributes *organized in one or multiple aggregation levels*. The definition of *dimension schema*s should be generic enough to cooperate with different DW and LOD models containing (a) multiple attributes within a level, (b) mono-hierarchical dimensions, (c) multiple-hierarchical dimensions and (d) non hierarchical dimensions.

Definition 1. A *dimension schema is a triple* $D_i=\langle \mathrm{dn}^i, \mathcal{L}^i, \prec^i \rangle$, such as:

- dn^i is the dimension's *name*;
- $\mathcal{L}^i$ is a finite set of pairs of $\langle l_x, \mathcal{A}_x \rangle$, such that l_x is a level and $\mathcal{A}_x$ is a set of attributes associating to a level;
  - among the levels, there exist at least a unique root level and a unique extreme level;
  - among the attributes of a level, the unique attribute identifying the level is called *parameter*, denoted p_x, while the other attributes describing the semantics of the *parameter* are called *weak attribute*, denoted $Weak_x$, $Weak_x=\mathcal{A}_x \backslash \{p_x\}$. The domain of an attribute a_k ($a_k \in \mathcal{A}_x$) is denoted as $dom(a_k)$.
- $\prec^i$ is a set of *asymmetric* and *transitive binary relation*s which reveals the aggregation path between a pair of levels. Remember that the asymmetry means that $\exists l_m, l_n, l_k \in \mathcal{L}^i : (l_m \prec^i l_n) \wedge (l_n \prec^i l_m) \Rightarrow l_m = l_n$, while the transitivity means that $(l_m \prec^i l_n) \wedge (l_n \prec^i l_k) \Rightarrow l_m \prec^i l_k$.

Remark. In the case of a non hierarchical dimension (e.g. dimensions in a QB schema), the root level contains all non-extreme attributes, while the set of *binary relation*s only includes the one between the root level and the extreme level.

We propose the following graphical notation of *dimension schema* for non-expert users (cf. Fig. 5). For simplicity, the extreme level is omitted in the graphical notation.

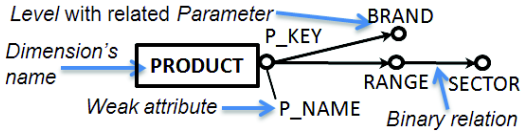


Fig. 5. Graphical notation of a *dimension schema*.

In the remainder of this paper, we abusively note $\mathcal{L}$ and $\prec$ instead of $\mathcal{L}^i$ and $\prec^i$ when it is clear that $\mathcal{L}$ and $\prec$ represents the element of the dimension D in question.

B. Dimension Instance

The conceptual definition of *dimension instance*s is required to ensure the interoperability between schema-centered DW models and instance-centered LOD models. A

dimension instance (a) maps the parameter at each level with the corresponding parameter instances, (b) describes the hierarchical aggregation relationships between parameter instances and (c) associates the values of a weak attribute to the related parameter instance.

Definition 2. A *dimension instance* consists of:

- a function between an attribute a_k ($a_k \in \mathcal{A}_x$) at the level l_x ($l_x \in \mathcal{L}$) and its instances (i.e. any constant in $\bigcup_{l_x \in \mathcal{L}} \pi_{l_x}(D)$[4]);
- a *rollup* function revealing the *part-whole* relationship between a child attribute and a parent attributes, denoted $\Uparrow_{p_x}^{p_y}$, from p_x to p_y, such that $l_x \prec l_y$, $\Uparrow_{p_x}^{p_y}$: $dom(p_x) \rightarrow dom(p_y)$;
- an association function $\text{Map}_{p_x}^{Weak_x}$ mapping an instance of the parameter p_x to the values of its weak attributes $Weak_x$, such that $\text{Map}_{p_x}^{Weak_x}$: $dom(p_x) \rightarrow 2^{\bigcup_{a_k \in Weak_x} dom(a_k)}$.

C. Unified Cube Schema

A *Unified Cube schema* represents the multidimensional structure of data coming from one or multiple sources. Related dimensions from different sources are linked together, so that decision-makers can obtain new perspectives and more complete information during analyses.

Definition 3. An *Unified Cube schema* is denoted as ⟨ucn, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{ExtLink}$⟩, such as:

- ucn is the name of the *Unified Cube*;
- $\mathcal{D}=\{D_1;\ldots;D_n\}$ is a finite set of dimensions;
- $\mathcal{M}=\{m_1;\ldots;m_k\}$ is a finite set of numeric indicators called *measures*;
- $\mathcal{ExtLink}$ is a set of *extrinsic links* depicting the relevance between data. An *extrinsic link* is an inter-dimension mapping which associates together two disparate levels on different dimensions. The starting point of an *extrinsic link* is called *mapping level*, while the end point is called *mapped level*.

Remark. The definition of *Unified Cube schemas* is generic enough to represent the structure of data from one or multiple sources. In the case where only one data source is involved, we obtain a *Unified Cube* without *extrinsic links* (i.e. $\mathcal{ExtLink}=\varnothing$). If multiple sources are included in a *Unified Cube*, a set of *extrinsic links* should be built to associate relevant data together. Details about different types of *extrinsic link* will be discussed in the section IV.B.

Fig. 6 shows the graphical notation of *Unified Cube* built upon the warehoused data. Measures sharing the same related dimensions are grouped together within the graphical notation.

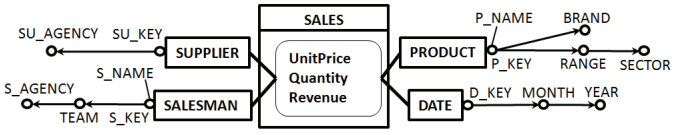


Fig. 6. Graphical notation of a *Unified Cube* with only warehoused data.

D. Unified Cube Instance

A *Unified Cube instance* (a) associates each measure value to the instances of parameters on the related dimensions and (b) connects the parameter instances at the mapping level with relevant ones at the mapped level.

Definition 4. Let $\{p_1,\ldots, p_n\}$ be a set of parameters, in which p_x is a parameter of the dimension D_x, $dom(m_j)$ is the values of the measure m_j, a *Unified Cube instance* consists of:

- a function between a set of parameter instances and the related measure values, such as $\forall m_j \in \mathcal{M}: 2^{\bigcup_{x\in[1,n]} dom(p_x)} \rightarrow dom(m_j)$;
- an instance of *extrinsic link* revealing the relevance between data. Each extrinsic link instance associates the parameter instances at a mapping level l_i with the relevant parameter instances at a mapped level l_j, such as $dom(p_i) \rightarrow dom(p_j)$[5].

IV. Process of Building a Unified Cube

Based on the generic multidimensional modeling language of *Unified Cubes*, warehoused data and LOD can be unified together in a single schema. In this section, we present a two-stage process allowing building a *Unified Cube* from multiple sources in the domains of DW and LOD.

As is shown in Fig. 7, the first step is to transform data schemas published with specific modeling vocabularies into a generic conceptual representation with a common modeling language. The multidimensional modeling language of *Unified Cube* can be used in the step 1, since it is generic enough to cover all the features embedded in multidimensional models. The conceptual representation obtained after the first step is called *exportation cube* whose aim is to facilitate the combination of different schemas in a *Unified Cube*.

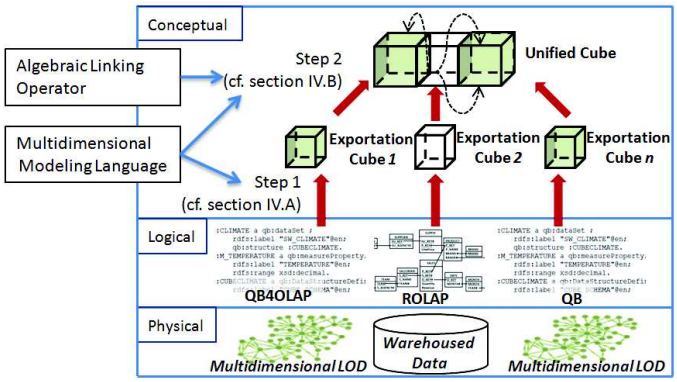


Fig. 7. Building a *Unified Cube* from multiple data sources.

The second step aims at linking together relevant data to obtain a unified schema. To do so, the multidimensional modeling language of *Unified Cube* provides a component, named *extrinsic link*, which allows representing the relevance between two dimensions. To facilitate the linking operations for non-expert users, we propose a high-level declarative language in the form of algebraic operator which translates a

[4] π represents the *projection* operator of the relational algebra

[5] σ represents the *selection* operator in the relational algebra

decision-maker's need into a mapping between two relevant dimensions. An algorithm is also proposed for the linking operator to automate its execution and guarantee the overall validity of a *Unified Cube* .

The outcome of the process is a *Unified Cube* which represents all useful data for analyses at the conceptual level along with the relevance between different dimensions.

A. Step I: Generic Conceptual Representation

Warehoused data and LOD are published according to different models. Each data model follows a specific modeling language expressed with the terminology of the corresponding domain. The differences between various modeling languages in DW and LOD domains make it impossible to analyze all useful data in a unified way. To overcome this problem, different data sources should be firstly transformed into a generic representation with a common modeling language.

The modeling language of *Unified Cube* is rich enough in expressivity to cover the complete multidimensional features embedded in OLAP, QB and QB4OLAP schemas (cf. section III). As the first step of building a *Unified Cube*, different data sources are transformed into a generic conceptual representation, named *exportation cube*, which aims at facilitating the combination of all useful data in a *Unified Cube*. No user intervention is needed for the first step. An *exportation cube* is automatically obtained by referring to the translation rules presented in this section.

1) From an OLAP Schema to an Exportation Cube

The transformation of an OLAP schema into a conceptual representation has been thoroughly studied during the last decade [5]. Since the modeling language of *Unified Cube* covers all multidimensional features of OLAP models, the transformation of an OLAP schema into *exportation cube* is quite straightforward and thus does not need to be repeated in this paper. The graphical notations of the obtained *exportation cube* of the R-OLAP DW (cf. Fig. 1) can be found in Fig. 6.

2) From a QB4OLAP Schema to an Exportation Cube

As an extension of QB vocabulary, the main idea of QB4OLAP is to use RDF triples to represent the most common features of a multidimensional model. More specifically, QB4OLAP adds classes and properties (prefixed by *qb4o*) to represent multiple attributes (i.e. a parameter and its weak attributes) within a level, hierarchical dimensions with multiple levels and the set of aggregate functions associated to a measure. TABLE I. gives an overview of the conceptual definitions of *exportation cube*s and their corresponding QB4OLAP triples. Note that in the domain of LOD, a variable starts with a question mark.

TABLE I. UNIFIED CUBE CONCEPTS AND QB4OLAP TRIPLES

Unified Cube Concept	QB4OLAP triples
Dimension schema (?dim)	?dim `a qb:DimensionProperty`;
Level (?lvl)	?dim `qb4o:hasHierarchy` ?hierarchy; ?hierarchy `a qb4o:hierarchyProperty`; ?hierarchy `qb4o:hasLevel` ?lvl.
Parameter (?para) weak attribute (?att)	?lvl `a qb4o:levelInHierarchy`; ?lvl `qb4o:levelComponent` ?para; ? para `a qb:levelProperty`; ?para `qb4o:hasAttribute` ?att.
Binary Relation (?biRel)	?biRel `a qb4o:hierarchyStep`; ?biRel `qb4o:childLevel` ?lvlChild; ?biRel `qb4o:parentLevel` ?lvlParent; ?biRel `qb4o:cardinality` ?cardinalityParentChild
Dimension Instance (?dimIns, ?dimInsParent	?dimIns `qb4o:inLevel` ?lvlChild; ?dimIns `rdfs:label` ?InsChild; ?dimIns `skos:broader` ?dimInsParent. ?dimInsParent `qb4o:inLevel` ?lvlParent; ?dimInsParent `rdfs:label` ?InsParent.
Cube Schema (?cube)	?cube `a qb:dataSet`; ?cube `rdfs:label` ?cubeName; ?cube `qb:structure` ?cubeStructure; ?cubeStructure `a qb:DataStructureDefinition`; ?cubeStructure `qb:component [qb4o:level` ?lvlRoot; `qb4o:cardinality` ?cardinalityCubeDim`]`; ?cubeStructure `qb:component [qb:measure` ?measure`]`.
Measure (?measure)	?measure `a qb:measureProperty`; ?measure `rdfs:label` ?mName.
Cube Instance (?ob)	?ob `a qb:Observation`; ?ob `qb:dataSet` ?cube; ?ob ?dim ?dimIns; ?ob ?measure ?value.

Based on the TABLE I. we translate the LOD dataset about the outlet sales into an *exportation cube*. Its graphical notations are shown in Fig. 8.

3) From a QB Schema to an Exportation Cube

QB is the current W3C standard to publish statistical LOD. In terms of multidimensional features, QB allows defining the structure of a fact via `qb:DataStructureDefinition`. However, it does not support the same hierarchical structure of dimension as a multidimensional model does. In a multidimensional model, a *dimension schema* corresponds to a hierarchical structure whose attributes are organized according to multiple granularity levels. A *dimension schema* defined in QB, on the contrary, may only contain one granularity level with a non-hierarchical list of attributes.

One way to find the hierarchical relationship between two attributes within a QB dimension is to look into *dimension instances*. However, obtaining a hierarchical *dimension schema* from a QB dataset is not straight-forwards as it seems to be. For instance, the property `skos:hasTopConcept` (or `qb:hierarchyRoots` in the case of non-skos hierarchies) is, by convention, used to link a *dimension schema* to the *topmost* attribute in a hierarchy. However, on the contrary to a hierarchy in a multidimensional schema, there is no integrity constraint enforcing the *top-most* convention in a QB schema. By consequence, publishing the following *dimension schema*, which is strictly forbidden according to a multidimensional model, is nevertheless consistent with the QB specifications.

```
eg:dimSchema skos:hasTopConcept eg:oneAtt
eg:oneAttribute skos:broader eg:anotherAtt
eg:anotherAtt skos:inScheme eg:dimSchema
```

Facing these issues, we propose the following algorithm to obtain an *exportation cube* from a QB dataset.

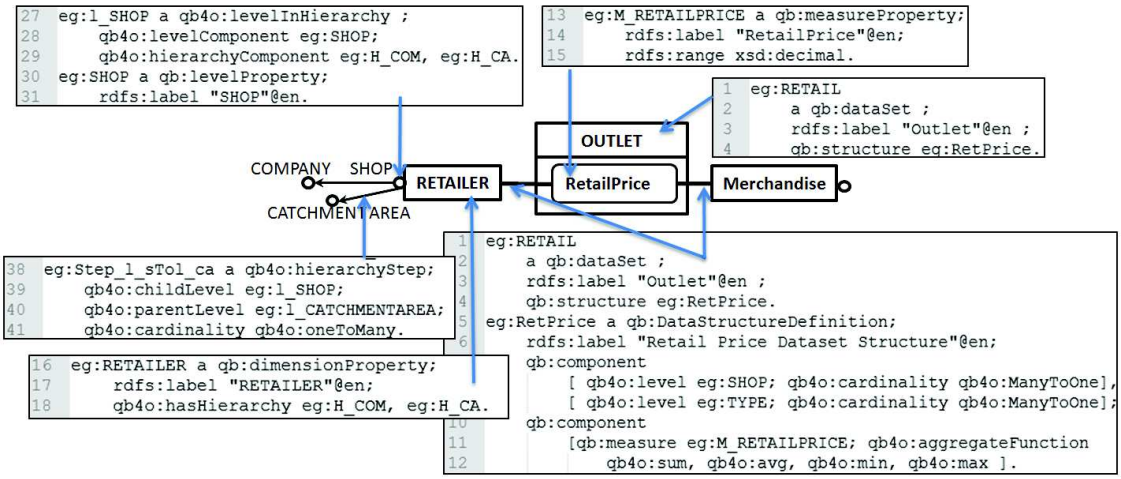


Fig. 8. *Exportation Cube* for LOD in QB4OLAP format.

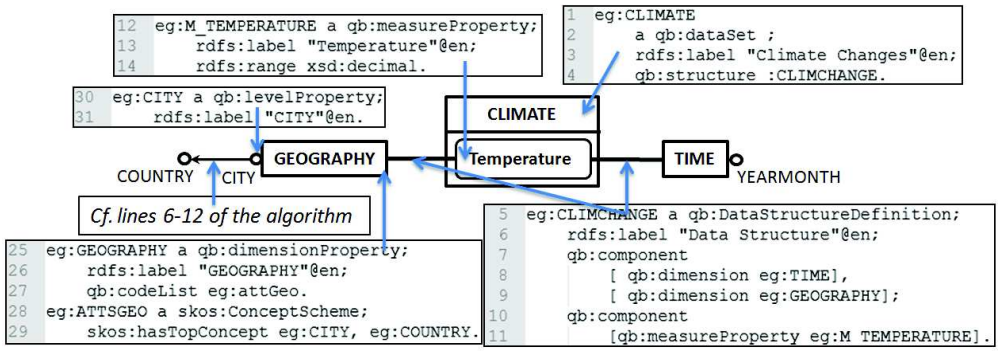


Fig. 9. *Exportation Cube* for LOD in QB4OLAP format.

Algorithm Creating an *exportation cube* from a QB schema
Input: A QB schema *qbCube* where *cubeStructure* is the stucture of the schema; D_{QB} is the set of dimensions; M_{QB} is the set of measures.
Output: A conceptual *exportation cube*.
Begin
1. For each $d_i \in D_{QB}$ (d_i `a qb:dimensionProperty`)
2. Create a dimension with the name *dimN* such as d_i `rdfs:label` *dimN*;
3. For each attribute a_k of d_i (d_i `qb:codeList` cl_i; cl_i `a skos:ConceptScheme`; cl_i `skos:hasTopConcept` a_k)
4. Create a level l_k for a_k;
5. Associate $\langle l_k, a_k \rangle$ to d_i;
6. For **all** attribute instances I_k of a_k (I_k `a` a_k)
7. If I_k `skos:broader` I_{k+1} (I_{k+1} `a` a_{k+1})
8. Create a *binary relation* $l_k \prec l_{k+1}$;
9. ElseIf I_k `skos:narrower` I_{k-1} (I_{k-1} `a` a_{k-1})
10. Create a *binary relation* $l_{k-1} \prec l_k$;
11. End If
12. End for
13. End For
14. End For
15. Create an *exportation cube* named *cubeN* such as *qbCube* `a qb:dataSet`; *qbCube* `rdfs:label` *cubeN*;
16. For each $m_j \in M_{QB}$ (m_j `a qb:measureProperty`)
17. Create a measure in the *exportation cube* with the name *meN* such as m_j `rdfs:label` *meN*;
18. Associate this measure with its associated dimensions $D_{mj} \subseteq D_{QB}$, such as $\forall d_x \in D_{mj}$: *cubeStructure* `a qb:DataStructureDefinition`; *cubeStructure* `qb:component [qb:dimension` d_x `], qb:component [qb:measureProperty` m_j`] ;`
19. End for
End.

Remark. As the authors of [6] mention, the QB format is not always correctly used in some real-world application cases. To avoid bringing inaccurate information to decision-makers, error-detection methods such as those presented in [7] should be applied to a QB schema before applying the proposed algorithm.

We apply the algorithm to the QB dataset about climate changes. The obtained *exportation cube* is shown in Fig. 9. After each data source is transformed into a corresponding *exportation cube*, we regroup together all *exportation cube*s in a preliminary *Unified Cube*. The preliminary *Unified Cube* aims at providing a unified view of all useful data for analyses, so that decision-makers can associate relevant data together during the second step of the process.

B. Step II: Cube Linking

The multidimensional modeling language of *Unified Cube*s includes relationships between data within one source and inter-schema relationships representing the relevance

between data from disparate sources (i.e. *extrinsic links*). In the previous sections, we discuss how different relationships embedded in one data source can be automatically transformed into *exportation cubes*. The aim of this section is to present how inter-schema relationships are established and managed in a *Unified Cube*, which corresponds to the second step of the process. First, we describe two types of *extrinsic links* that a decision-maker can define between two related dimensions. Second, we propose a high-level declarative language in the form of algebraic operator, named *DLink*, which builds *extrinsic links* between relevant data according to decision-makers' needs. An algorithm is proposed to automate the execution of the *DLink* operator while ensuring the validity of the built *Unified Cube*.

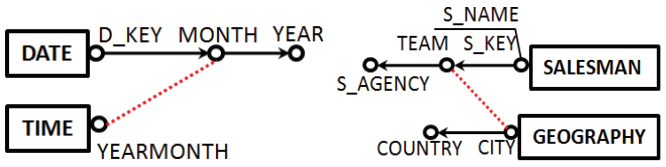
1) Types of Extrinsic Links

The relevance between data from disparate sources may take many forms. Among different types of relevance, the correlation and the aggregation relations are particularly useful in the context of business analysis.

a) Correlation Links

Classically an analysis granularity includes only one level of dimension. In the context of *Unified Cubes*, the notion of analysis granularity needs to be generalized, since several levels of dimension from different data sources may refer to the same analysis granularity. To this end, we propose *correlation links* which associate together two relevant levels that belong to the same analysis granularity.

A *correlation* link allows associates a pair of levels on two dimensions that are considered equivalent in term of analytical needs. The parameters involved in the levels associated by a *correlation* link may be semantically equivalent (i.e. referring to the same concept), such as the parameter named *YearMonth* in the QB dataset and the one named *Month* in the R-OLAP DW (cf. Fig. 10(a)). In the graphical notation of a *Unified Cube*, *correlation* links are represented by dotted lines.



(a) Semantically equivalent elements (b) Analytically equivalent elements

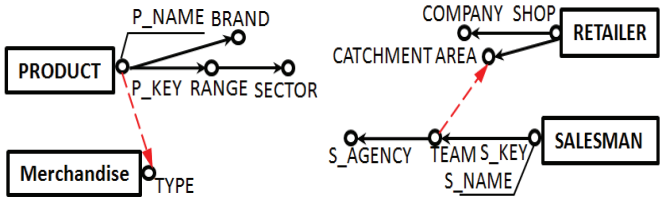
Fig. 10. C*orrelation* links at the schema level.

Two levels that are semantically heterogeneous but share the same role in business analyses (i.e. analytically equivalent) may also be connected together through a *correlation* link. For instance, the salesmen's *team* and the *city* are two semantically heterogeneous concepts. However, in a specific analysis context where each *team* is in charge of sales in one *city*, analyses involving a *team* may as well be carried out with the corresponding *city*. In this case, the salesmen's *team* and the *city* can be considered as two analytically equivalent concepts, which can also be connected together through a *correlation* link (cf. Fig. 10(b)).

b) Aggregation Links

Classical multidimensional schemas allow decision-makers to carry out analyses only according to aggregation paths situated within one dimension. Within a *Unified Cube*, a level within a dimension may be aggregated to another level on a different dimension. To provide additional aggregation paths for analyses, we propose *aggregation* links which reveal the parent-child relation between two relevant levels on different dimensions.

An *aggregation* link gathers parts into a whole by associating one or several instances of the parameter at the mapping level to at most one parameter instance at the mapped level. An *aggregation* link may be inferred through the semantics embedded in the corresponding parameters. For instance, in Fig. 11(a) since each *Type* of *merchandise* may correspond to several *product* (i.e. *P_Key*), we can build an *aggregation* link between the levels l_{Type} and $l_{\text{P\_Key}}$. An *aggregation* link is represented by a dotted arrow in the graphical notation of a *Unified Cube*.



(a) Semantic *parent-child* relation (b) Analytical *parent-child* relation

Fig. 11. *Aggregation* links at the schema level.

In the previous case, the *aggregation* link represents the parent-child relation between a *product* and a *type* which can be deduced independently of the analysis context. An aggregation link may also be placed between two levels whose parent-child relations holds only under a given analysis context. For instance, the *aggregation* link in Fig. 11(b) between salesman's *team* and retailer's *catchment area* cannot be detected through the semantics embedded in the parameters. It is valid only under a specific analysis context where a *catchment area* of a retailer attracts the same clientele of the nearby salesman's *teams*.

2) Establishing Extrinsic Link

As one of the key characteristics of the next generation of BI, the internal complexity of a system should be hidden from end-users. In the context of a *Unified Cube*, it comes down to empower non-expert users with the ability to associate by themselves relevant data together according to their needs. Ideally, decision-makers could be able to express their needs through a high-level declarative language to a system. To this end, we define an user-oriented linking operator presented in an algebraic form, named *DLink*. The *DLink* operator allows associating relevant dimensions and producing a *Unified Cube* of all useful information as output. An algorithm is proposed to automate the execution of the *DLink* operator and ensure the validity of the obtained *Unified Cube*.

a) Algebraic Linking Operator

The *DLink* operator builds an *extrinsic link* according to a specified *type* between a pair of *levels* on two *dimensions*. TABLE II. shows the algebraic representation of *DLink*.

TABLE II. ALGEBRAIC DLINK OPERATOR

DLink ($D_{mapping}$; $l_{mapping}$; D_{mapped}; l_{mapped}; Type)=Unified Cube	
Input	- $D_{mapping}$: the starting dimension of the *extrinsic link*; - $l_{mapping}$: the starting level of the *extrinsic link*; - D_{mapped}: the ending dimension of the *extrinsic link*; - l_{mapped}: the ending level of the *extrinsic link*; - Type={*correlation link*; *aggregation link*}: the type of extrinsic link to establish between levels.
Output	Unified Cube: a *Unified Cube* with an updated set of *extrinsic link*s.

At the schema level, the *DLink* operator associates a pair of levels with a *correlation* link or an *aggregation* link. At the instance level, different mapping functions are created between a parameter instance at the *mapped* level and the corresponding one(s) at the *mapping* level.

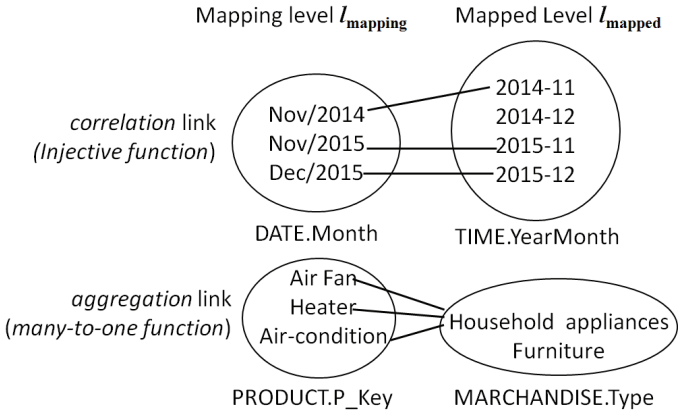


Fig. 12. Extrinsic links at the schema level and mappings at the instance level.

As is shown in Fig. 12, the *correlation* link corresponds to an *injective* function between the *mapping* level and the *mapped* level. The injection implies an instance of the parameter at the level $l_{mapping}$ is mapped to at most one instance of the parameter at the level l_{mapped}. The *correlation* link in Fig. 12 is built via the following operator: ***DLink***(D_{DATE}; l_{Month}; D_{TIME}; $l_{YearMonth}$; {*correlation link*}).

The *aggregation* link is a *many-to-one* function between the mapping level and the mapped level. A *many-to-one* function associates one or several child instances at the level $l_{mapping}$ to at most one parent element at the level l_{mapped} (cf. Fig. 12). For instance, the *aggregation* link associating one or several *product*s to their corresponding *type* is built through the following operator: ***DLink***($D_{PRODUCT}$; l_{P_Key}; $D_{MARCHANDISE}$; l_{Type}; {*aggregation link*}).

b) Execution Algorithm of the DLink Operator

To facilitate the tasks of non-expert users, we propose an algorithm to automate the execution of the *DLink* operator. Two types of operations are included in the algorithm, namely dimension mapping and validity verification. The dimension mapping operations aim at creating automatically or semi-automatically an *extrinsic link* between a pair of relevant levels on two dimensions, while the validity verification operation aim at removing inconsistent or redundant *extrinsic link*s in the new *Unified Cube*. The execution algorithm is as follows.

Algorithm Execution of *DLink*

Input: $\mathbf{D_{mapping}}$, $\boldsymbol{l_{mapping}}$: the starting dimenison and level of the new *extrinsic link*; $\mathbf{D_{mapped}}$, $\boldsymbol{l_{mapped}}$: the ending dimenison and level of the new *extrinsic link*; ***Type***: the type of the new *extrinsic link* to build.

Output: A ***Unified Cube*** with an updated set of *extrinsic link*s.

Begin

1. If the chosen ***Type*** *is* ***correlation link***
2. If *there exists an aggregation link between* $\boldsymbol{l_m}$ *and* $\boldsymbol{l_n}$ ($\boldsymbol{l_m} \in \mathcal{L}_{mapping} \wedge \boldsymbol{l_m} \prec \boldsymbol{l_{mapping}}$, $\boldsymbol{l_n} \in \mathcal{L}_{mapped} \wedge \boldsymbol{l_{mapped}} \prec \boldsymbol{l_n}$)
3. *Remove the aggregation link between* $\boldsymbol{l_m}$ *and* $\boldsymbol{l_n}$;
4. End if
5. *Create a correlation link between* $\boldsymbol{l_{mapping}}$ *and* $\boldsymbol{l_{mapped}}$;
6. Else If *the chosen* ***Type*** *is* ***aggergation link***
7. If *there exists an aggregation link between* $\boldsymbol{l_m}$ *and* $\boldsymbol{l_n}$ ($\boldsymbol{l_m} \in \mathcal{L}_{mapping} \wedge \boldsymbol{l_m} \prec \boldsymbol{l_{mapping}}$, $\boldsymbol{l_n} \in \mathcal{L}_{mapped} \wedge \boldsymbol{l_{mapped}} \prec \boldsymbol{l_n}$)
8. *Remove the aggregation link between* $\boldsymbol{l_m}$ *and* $\boldsymbol{l_n}$;
9. End If
10. If *there exists an extrinsic link (correlation or aggregation) between* $\boldsymbol{l_p}$ *and* $\boldsymbol{l_q}$ ($\boldsymbol{l_p} \in \mathcal{L}_{mapping} \wedge \boldsymbol{l_{mapping}} \prec \boldsymbol{l_p}$, $\boldsymbol{l_q} \in \mathcal{L}_{mapped} \wedge \boldsymbol{l_q} \prec \boldsymbol{l_{mapped}}$)
11. D*isplay a warning message*: *Impossible to place an aggregation link between* $\boldsymbol{l_{mapping}}$ *and* $\boldsymbol{l_{mapped}}$;
12. Else
13. *Create an aggregation link between* $\boldsymbol{l_{mapping}}$ *and* $\boldsymbol{l_{mapped}}$;
14. End If
15. End If

End

The dimension mapping operations establish a new *extrinsic link* between relevant dimensions (cf. lines 5 and 13). During the dimension mapping operations, relevant dimensions are associated together at both schema level and instance level. If the two dimensions correspond semantically (e.g. *Date* and *Time*, *Product* and *Merchandise*), the execution of dimension mapping operations may be automated. In this case, we may refer to the approaches presented in [8] to automatically establish mappings at the schema level as well as instance level. When two dimensions are relevant only under certain analysis contexts (e.g. *Salesman* and *Retailer*, *Salesman* and *Geography*), a semi-automatic approach should be adopted. In this case, the system proposes to decision-makers several possibilities of mappings between two dimensions based on domain ontology, semantic annotations or integrity constraints [9], [10]. Then decision-makers can choose or modify a proposed mapping according to a given analysis context.

Through the dimension mapping operations, decision-makers can associate, according to their analysis needs, relevant dimensions within a *Unified* C*ube*. At the same time, the algorithm of the *DLink* operator must make sure the global validity of all *extrinsic link*s in the new *Unified Cube*, especially when each new *extrinsic* link seems to be valid if being built individually. To do so, the algorithm carries out

validity verification operations to check the validity of the new *extrinsic link* and the existing ones. Decision-makers do not need to worry about the order in which the set of *extrinsic link*s should be placed, because inconsistent or redundant *extrinsic link*s are automatically removed from the *Unified Cube* after the validity verification operations.

More specifically, the algorithm checks if the new *extrinsic* link can replace a redundant *extrinsic link* already existing in a *Unified Cube*. Firstly, a *correlation* link can substitute an *aggregation* link built with a lower mapping level and/or a higher mapped level (cf. lines 2-4 of the following algorithm). For instance, suppose there was an *aggregation* link between the level $l_{\text{YearMonth}}$ of the dimension *TIME* and the level l_{Year} on the dimension *DATE*. Once the *correlation* link between $l_{\text{YearMonth}}$ and l_{Month} (cf. Fig. 10(a)) is built, the *aggregation* link between $l_{\text{YearMonth}}$ and l_{Year} should be removed from the *Unified Cube*, since it can be deduced from the new *correlation* link along with the *binary relation* $l_{\text{Month}} \prec l_{\text{Year}}$.

Similarly, an *aggregation* link can replace another *aggregation* link built with a lower mapping level and/or a higher mapped level (cf. lines 7-9 of the following algorithm). For instance, building the *aggregation* link between l_{Team} and $l_{\text{CatchmentArea}}$ would replace an *aggregation* link between $l_{\text{S\_Key}}$ and $l_{\text{CatchmentArea}}$, because the latter can be deduced from the new *aggregation* link and the *binary relation* $l_{\text{S\_Key}} \prec l_{\text{Team}}$ (cf. Fig. 11(a)).

On the other hand, a new *aggregation* link should not be built if a *correlation* link or an *aggregation* link already exists at a higher mapping level and/or a lower mapped level in the same *Unified Cube* (cf. lines 10 and 11 of the algorithm). For instance, since there is already a valid *correlation* link between l_{City} and l_{Team} (cf. Fig. 10(b)), no more *aggregation* link is allowed between l_{City} and $l_{\text{S\_Agency}}$ since it can be deduced from the *correlation* link and the *binary relation* $l_{\text{Team}} \prec l_{\text{S\_Agency}}$.

After the execution of four *DLink* operators, two *correlation* links and two *aggregation* links are built within the *Unified Cube* as shown in the Fig. 13.

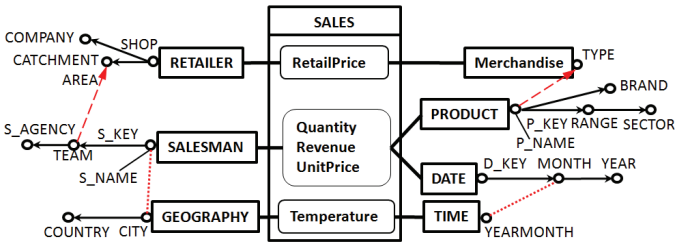


Fig. 13. *Unified Cube* for warehoused data and LOD

V. Experimental Assessments

In this section, we carry out some experimental assessments to demonstrate the feasibility of analyzing both warehoused data and LOD in a unified way. The advantage of unified analyses is that decision-makers can obtain complete and coherent information about an analysis subject and new analysis possibilities by following the relationships established between relevant data from different sources.

A. Protocol

1) Configuration for Data Implementation

To simulate the distributed nature of warehoused data and LOD, we use three identical Microsoft Windows 7 work stations located on the same LAN for the experimental assessments (Interl(R) i7-4510U 2GHz CPU, 8GB RAM, SSD 500GB disk). Each work station hosts one source of the running example: the DBMS Oracle 11g manages the warehoused data implemented with a snowflake schema, while a native triple store with an integrated SPARQL end-point provided by Apache Jena API [11] is used for LOD published in QB and QB4OLAP formats. Several warm-up runs are carried out in each work station after the installation. We implement the *Unified Cube* in a virtual machine installed on the same work station hosting the R-OLAP DW. Only one dedicated core of CPU, 500M RAM and 1GB disk are allocated to the virtual machine, since managing a *Unified Cube* is not resource-consuming; it only requires hosting a non-materialized view for the *Unified Cube* and an ontology implementing the set of *extrinsic link*s by associating together relevant data at the instance level.

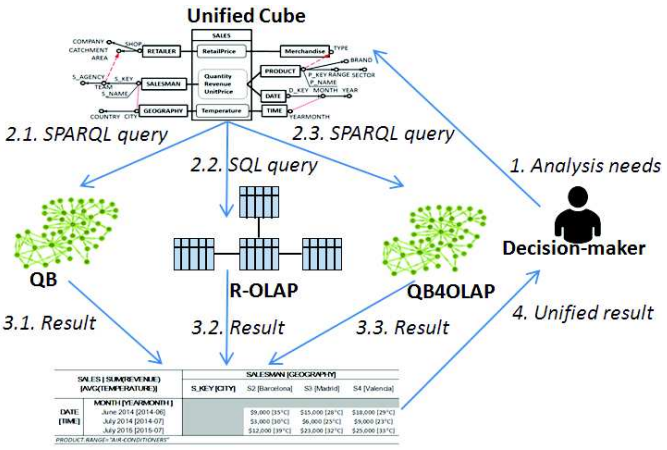


Fig. 14. Procedures for processing analyses in a *Unified Cube*.

An analysis need over a *Unified Cube* is translated into one or several queries which are applied to the corresponding component data sources. The analysis processing procedures are as follows (cf. Fig. 14). First, by identifying all useful measures and attributes in a *Unified Cube*, a decision-maker expresses an analysis need. This analysis need is manually divided into a set of user-oriented algebraic operators. Each algebraic operator involves data from one source. Two existing tools with graphical interface developed within our research team [12], [13] are used to automate the translation of an algebraic operator into an executable query over a R-OLAP DW or a LOD dataset. Then, each of the three work stations receives a SQL or SPARQL query and produces a partial result based on the hosted source. Each partial result is cleaned up, especially for SPARQL queries in which the URI of each returned triple is simplified to keep only useful information for analyses. At last, we query the ontology implementing the *extrinsic link*s to associate relevant data scattered among the partial results. In this way, a unified result is created before being presented to decision-makers.

2) Queries and Data Collection

To support analyses over both warehoused data and LOD, different types of queries should be correctly generated and executed in a *Unified Cube*. We carry out analyses covering the most widely used types of queries in the context of multidimensional analyses, i.e. queries containing joins, projections, selection criteria and aggregation functions with grouping clauses (i.e. `Group By`). More details are available in TABLE III.

TABLE III. ANALYSIS NEEDS, QUERY TYPES AND DATA SOURCES

Analysis Need	Query	Source
1. Find the sale *quantities* for *heater*s (product range) according to the *country* in which the salesmen promote.	Join, Select	R-OLAP QB
2. Find the average *price* and *quantity* of products sold by each salesman's *team*, associating the *price* with the average outlet *retail price* for the products of the same *type* offer by the *retailer*s in the corresponding *catchment area*.	Join, Selection, Projection Group By	R-OLAP, QB4OLAP

In consideration of the computing capacity of the three work stations, we populate the three datasets with a reasonable amount of synthetic data to avoid query execution timeout. The R-OLAP DW contains over 16 million tuples while the QB and QB4OLAP schemas respectively include about 1.44 million and 1.21 million triples[6].

B. Result and Discussion

In this section, we present and discuss the results of our experimental assessments. To separate the fundamental operations from the complex syntax of querying languages, the generated SQL and SPARQL queries are presented in the form of relational algebra and SPARQL algebra[7] respectively. Note that although the SPARQL algebra has not yet become a W3C standard, it is already supported by several frameworks for querying LOD: all algebraic expressions of SPARQL queries in this section are generated by Apache Jena API.

The first analysis corresponds to a cross-source analysis which calculates measures from one data source according to parameters in another data source. During this analysis, the *Unified Cube* offers additional perspectives of analysis to decision-makers by aggregating sale *quantities* from R-OLAP DW according to the parameter named *Country* from the QB dataset. This cross-source analysis is feasible within the *Unified Cube* owning to the *correlation* link between l_{S_Key} on $D_{Salesman}$ and l_{City} on $D_{Geography}$ as well as the *binary relation* $l_{City} \prec l_{Country}$ within the dimension *Salesman*.

Two queries are generated for the first analysis (cf. Fig. 15). The SQL query searches for sale quantities of heaters by salesmen, while the SPARQL query returns the list of countries in the QB dataset.

[6] The source code and generated queries are available at *http://olap-sw.weebly.com/blog/rcis-2016-source-code-and-queries*

[7] https://www.w3.org/2001/sw/DataAccess/rq23/rq24-algebra.html

```
S_Key 𝓕sum(quantity)((Sales JOIN P_Key= P_Key
  SELECTRange='Heaters'(Product)))
```

(a) *SQL Query*

```
(PROJECT (?country)
  (BGP
    (TRIPLE ?geo a qb:dimensionProperty)
    (TRIPLE ?geo rdfs:label "Geography"@en)
    (TRIPLE ?geo qb:codeList ?lstP)
    (TRIPLE ?lstP skos:hasTopConcept ?pCountry)
    (TRIPLE ?pCountry a qb:levelProperty)
    (TRIPLE ?pCountry rdfs:label "Country"@en)
    (TRIPLE ?insCountry a ?pCountry)
    (TRIPLE ?insCountry rdfs:label ?country)))
```

(b) *SPARQL Query*

Fig. 15. Generated SQL and SPARQL queries for the first analysis.

After querying the ontology implementing the *extrinsic link*s, relevant instances of parameters *S_Key* and *City* are associated together. Then, by referring to the *rollup* function between the instances of *City* and *Country*, a country is associated with a corresponding group of salesmen within the unified analysis result which regroups the results of SQL and SPARQL queries. Through the result shown in Fig. 16, the decision-maker realizes that the salesmen who have difficulty in selling the heaters are in fact located in hot countries where the demand of heaters remains relatively low. Without the *Unified Cube*, it would be difficult to obtain multiple perspectives in such an intuitive way if the analysis was carried out with separate data sources.

Parameters from QB dataset *Measures from R-OLAP DW*

GEOGRAPHY [SALESMAN]	SALES
COUNTRY [S_KEY]	SUM(QUANTITY)
Finland [S1; S7; S9]	121
Spain [S2;S3; S4]	51
Sweden [S5; S6; S8]	158
PRODUCT.RANGE="HEATERS"	

Fig. 16. Unified result of the first analysis.

The second analysis corresponds to a comparative analysis which requires including relevant information from different data sources in one analysis result. A *Unified Cube* allows displaying measures from different sources according to related dimensions which are interconnected through *extrinsic link*s. During the second analysis, the measures *quantity* and *unit price* from the R-OLAP DW can be displayed along with the *retail price* from the QB4OLAP dataset starting from the levels associated by the *aggregation* link between $D_{Product}$ and $D_{Merchandise}$ as well as the one between $D_{Salesman}$ and $D_{Retailer}$.

The second analysis need is translated into a SQL query and a SPAQL query (cf. Fig. 17). The SQL query searches for the *unit prices* and the sale *quantities* by *products* and salesman' *teams*, while the SPARQL query reveals the *retail price* of the same product's *type* offered by retailers in the *catchment areas* sharing the same clientele of salesman's *team*s.

Team, P_Key $\mathcal{F}_{avg(unitPrice),\ avg(quantity)}$ ((Sales JOIN$_{S_KEY=S_KEY}$ Product)

(a) *SQL Query*

```
(PROJECT (?catchmentArea ?type ?RetailPrice)
 (EXTEND ((?RetailPrice ?.0))
  (GROUP (?catchmentArea ?type) ((?.0 (avg ?price)))
   (SEQUENCE
    (BGP
     (TRIPLE ?obs qb:dataSet ?ds) (TRIPLE ?obs a qb:observation)
     (TRIPLE ?ds rdfs:label "Outlet"@en)
     (TRIPLE ?obs ?measure ?price) (TRIPLE ?measure a qb:measureProperty)
     (TRIPLE ?measure rdfs:label "RetailPrice"@en)
     (TRIPLE ?obs ?retailer ?shop)
    )
    (PATH ?shop (path* skos:broader) ?ca)
    (BGP
     (TRIPLE ?ca rdfs:label ?catchmentArea)
     (TRIPLE ?ca qb4o:inLevel ?lvlCA)
     (TRIPLE ?lvlCA qb4o:levelComponent ?pCA)
     (TRIPLE ?pCA rdfs:label "Catchement Area"@en)
     (TRIPLE ?obs ?merchandise ?mType)
     (TRIPLE ?mType rdfs:label ?type)
     (TRIPLE ?mType qb4o:inLevel ?lvlType)
     (TRIPLE ?lvlType qb4o:levelComponent ?pType)
     (TRIPLE ?pType rdfs:label "Type"@en))))))
```

(b) *SPARQL Query*

Fig. 17. Generated SQL and SPARQL quereis for the second analysis.

After the execution of these two partial queries, each product is associated with its corresponding type through the *aggregation* link between l_{P_Key} on $D_{Product}$ and l_{Type} on $D_{Merchandise}$. Meanwhile, via the *aggregation* link between l_{Team} on $D_{Salesman}$ and $l_{CatchmentArea}$ on $D_{Retailer}$ each salesman's team is aggregated to the related catchment area which shares the same clientele of the company. The unification of relevant dimensions makes it possible to combine measures from R-OLAP DW with the one from QB4OLAP dataset. By comparing with the price offered by other retailers (cf. Fig. 18), the decision-maker notices that the higher the sale prices fixed by the company, the lower the sale quantities promoted by the salesmen. It would be less easy to obtain such a comprehensive analysis result without gathering relevant data in dispersed sources into a *Unified Cube*.

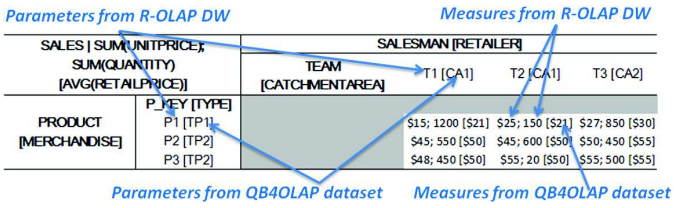


Fig. 18. Unified result of the second analysis.

We record the execution time of each analysis from the moment where an analysis need is submitted from the decision-maker till the moment where a unified result is produced by gathering all partial results. Without any optimization technique for query execution, all analyses return the result within one minute (cf. TABLE IV. column *Execution Time*). The execution time remains reasonable in consideration of the laptop-level configuration of the working stations.

TABLE IV. EXECUTION TIME OF EACH ANALYSIS

Analysis Need	Execution Time (Seconds)	Working Station's Response Time (Seconds)		
		R-OLAP	*QB*	*QB4OLAP*
1	29.8	7.9	27	n/a
2	56.5	8.5	n/a	51.0

A *Unified Cube* increases the efficiency of analysis by allowing decision-makers to analyze all useful data in a unified way. The execution time of each analysis is much lower than the sum of response time of all working stations, which is impossible if decision-makers query different data sources separately and wait for all partial results one after another. Meanwhile, each working station receives and executes a partial query individually without any influence on others. Thus, another advantage of *Unified Cube*s is the possibility of parallelizing the execution of partial queries, which helps further reduce the execution time of analysis.

VI. RELATED WORK

In today's highly dynamic business context, well-informed and effective decisions require enriching warehoused data with other data that fall outside of the scope of internal DWs [1], [14]. Establishing relationships between elements from different sources belongs to the field of *schema matching*. Being studied for the first time as an independent research topic in [15], schema matching refers to a set of techniques allowing generating correspondences between several formal structures that represent an engineered artifact. Within the scope of this paper (i.e. the fields of DW and LOD), the classical application cases of schema matching include ETL and ontology alignment [8].

In the DW domain, an ETL (*extract-transform-load*) process integrates one or several disparate data sources (e.g. operational databases) into the warehouse format. With the arrival of web published data in business analyses, the DW community intuitively treated LOD as external data sources that should be centralized in a DW through an ETL process [16], [17]. The shortcoming of this approach was found soon afterwards due to the poor freshness of warehoused LOD and the high cost of non-automatic ETL process [18]. To overcome these drawbacks, [6] and [19] propose solutions to querying directly LOD published in QB or other multidimensional formats derived from QB. Among the derived LOD formats, the QB4OLAP vocabulary allows representing the complete multidimensional structure of LOD at the logical level [4]. However, the previous work is limited to representing LOD from one data source with a multidimensional structure expressed through semantic web languages. No solution is proposed to deal with the problems about how disparate data sources can be represented and analyzed in a unified way. In this paper, we define a generic conceptual model allowing including both warehoused data and LOD. Differing from the semantic web approaches whose main goal is fixing a common vocabulary and a set of interpretation constraints (i.e., inferring rules) to describe the data semantics, our model aims at including all useful data in a generic and unified multidimensional schema.

In the field of semantic web, ontology alignment is the de facto application case of schema matching [20], [21]. Ontology alignment allows matching semantically relevant

elements in different schemas. In term of practical utility for business analyses, ontology alignment is not generic enough to represent common analytical relationships that are semantically irrelevant. For instance, in our running example the statement "a salesman's *team* is a equivalent concept to *city* in an analytical context about the sales of air-conditioners" cannot be represented through any standardized ontology property without violating the strict logical semantics of identity: by no means *team* and *city* may refer to the same concept in terms of semantic. Our work is more generic since both semantically and analytically relevant data can be associated together according to decision-makers' needs.

VII. Conclusion And Discussions

Our aim is to make full use of all relevant data to support effective and well-informed decisions. To this end, we define a generic multidimensional model, named *Unified Cube*, which allows including as much useful information as possible for analyses. The multidimensional modeling language of *Unified Cube*s includes both conceptual definitions and graphical notations.

To build a *Unified Cube*, we describe a two-stage process which unifies relevant data from disparate sources. As a first step, schemas published with specific modeling languages in the DW and LOD domains are transformed into a common conceptual representation named *exportation cube*. Defined through a generic multidimensional modeling language, *exportation cube*s allow facilitating the combination of useful data in a *Unified Cube*. The second step of building a *Unified Cube* is to associate relevant data according to decision-makers' needs. The multidimensional modeling language includes a set of mappings, named *extrinsic link*s, which allows modeling inter-dimension analysis granularities (i.e. *correlation* links) and parent-child relations between levels on different dimensions (i.e. *aggregation* links). An algebraic linking operator named *DLink* is proposed to enable non-expert users to establish *extrinsic link*s according to their needs. We propose an algorithm to automate the execution of the *DLink* operator while guaranteeing the overall validity of all *extrinsic links* within a *Unified Cube*.

To demonstrate the feasibility of the proposed concepts, we develop a prototype including a *Unified Cube* built from a R-OLAP DW and two LOD datasets published in QB and QB4OLAP formats. By translating analysis needs into queries applicable to *Unified Cube*s, we show how warehoused data and LOD can be analyzed in a unified way.

One of our ongoing research efforts is focused on an analysis framework compatible with *Unified Cube*s. Following an automatic approach of federated queries processing, this analysis framework aims at enabling non-expert users to carry out on-the-fly analyses including multiples data sources within a *Unified Cube*. We also intend to generalize the definition of *extrinsic link*s to represent more sophisticated analysis needs. For instance, an n-ary link will be defined to model the relevance between three or more dimensions. A more long-term objective is to study the influences of the materialization of source data over the efficiency of analyses carried out in *Unified Cube*s.

References

[1] A. Abelló, O. Romero, T. B. Pedersen, R. Berlanga, V. Nebot, M. J. Aramburu, and A. Simitsis, "Using Semantic Web Technologies for Exploratory OLAP: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 571–588, Feb. 2015.

[2] M. E. Zorrilla, J.-N. Mazón, Ó. Ferrández, I. Garrigós, F. Daniel, and J. Trujillo, Eds., *Business Intelligence Applications and the Web: Models, Systems and Technologies*. IGI Global, 2012.

[3] C. Ciferri, R. Ciferri, L. Gómez, M. Schneider, A. Vaisman, and E. Zimányi, "Cube Algebra: A Generic User-Centric Model and Query Language for OLAP Cubes," *Int. J. Data Warehous. Min.*, vol. 9, no. 2, pp. 39–65, 32 2013.

[4] L. Etcheverry, A. Vaisman, and E. Zimányi, "Modeling and Querying Data Warehouses on the Semantic Web Using QB4OLAP," in *Data Warehousing and Knowledge Discovery*, vol. 8646, Cham: Springer International Publishing, 2014, pp. 45–56.

[5] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Rec.*, vol. 26, no. 1, pp. 65–74, Mar. 1997.

[6] B. Kämpgen and A. Harth, "Transforming statistical linked data for use in OLAP systems," in *Proceedings of the 7th international conference on Semantic systems*, Graz, Austria, 2011, pp. 33–40.

[7] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer, "Detecting Errors in Numerical Linked Data Using Cross-Checked Outlier Detection," in *The Semantic Web – ISWC 2014*, vol. 8796, Cham: Springer International Publishing, 2014, pp. 357–372.

[8] E. Rahm, "Towards Large-Scale Schema and Ontology Matching," in *Schema Matching and Mapping*, Springer Berlin Heidelberg, 2011, pp. 3–27.

[9] A. DOAN and A. Y. HALEVY, "Semantic integration research in the database community: A brief survey.," *AI Mag.*, vol. 26, no. 1, pp. 83–94, 2005.

[10] G. Cabanac, C. Max, F. Ravat, and O. Teste, "Decisional Annotations: Integrating and Preserving Decision-Makers' Expertise in Multidimensional Systems," in *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*, IGI Global, 2009, pp. 65–81.

[11] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings of the 13th international World Wide Web conference*, New York, NY, USA, 2004, pp. 74–83.

[12] R. Saad, O. Teste, and C. Trojahn, "OLAP Manipulations on RDF Data following a Constellation Model," in *Proceedings of International Workshop on Semantic Statistics (SemStats 2013) collocated with International Semantic Web Conference (ISWC-2013)*, Sydney, 2013.

[13] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh, "Algebraic and Graphic Languages for OLAP Manipulations," *Int. J. Data Warehous. Min.*, vol. 4, no. 1, pp. 17–46, 2008.

[14] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazón, F. Naumann, T. Pedersen, S. B. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen, "Fusion Cubes: Towards Self-Service Business Intelligence," *Int. J. Data Warehous. Min.*, vol. 9, no. 2, pp. 66–88, 32 2013.

[15] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB J.*, vol. 10, no. 4, pp. 334–350, Dec. 2001.

[16] V. Nebot, R. Berlanga, J. M. Pérez, M. J. Aramburu, and T. B. Pedersen, "Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses," in *Journal on Data Semantics XIII*, vol. 5530, Springer Berlin Heidelberg, 2009, pp. 1–36.

[17] O. Romero and A. Abelló, "Automating multidimensional design from ontologies," in *international workshop on Data warehousing and OLAP*, 2007, pp. 1–8.

[18] L. Etcheverry and A. A. Vaisman, "Enhancing OLAP Analysis with Web Cubes," in *The Semantic Web: Research and Applications*, vol. 7295, Springer Berlin Heidelberg, 2012, pp. 469–483.

[19] L. Etcheverry, S. Gomez, and A. Vaisman, "Modeling and Querying Data Cubes on the Semantic Web," *ArXiv Prepr. ArXiv151206080*, 2015.

[20] P. A. Bernstein, J. Madhavan, and E. Rahm, "Generic schema matching, ten years later," *Proc. VLDB Endow.*, vol. 4, no. 11, pp. 695–701, 2011.

[21] J. Euzenat, *Ontology matching*, 2nd edition. New York: Springer, 2013.