Building Data Warehouses with Semantic Web Data

Victoria Nebot*, Rafael Berlanga

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec, 12071, Castellón, Spain
Phone: (+34) 964 72 83 67. Fax: (+34) 964 72 84 35

Abstract

The Semantic Web (SW) deployment is now a realization and the amount of semantic annotations is ever increasing thanks to several initiatives that promote a change in the current Web towards the Web of Data, where the semantics of data become explicit through data representation formats and standards such as RDF/(S) and OWL. However, such initiatives have not yet been accompanied by efficient intelligent applications that can exploit the implicit semantics and thus, provide more insightful analysis. In this paper, we provide the means for efficiently analyzing and exploring large amounts of semantic data by combining the inference power from the annotation semantics with the analysis capabilities provided by OLAP-style aggregations, navigation, and reporting. We formally present how semantic data should be organized in a well-defined conceptual MD schema, so that sophisticated queries can be expressed and evaluated. Our proposal has been evaluated over a real biomedical scenario, which demonstrates the scalability and applicability of the proposed approach.

Keywords: Semantic Web, Ontologies, Data Warehouses, OLAP, Multidimensional design

1. Introduction

The effort behind the Semantic Web (SW) is to add machine-understandable, semantic annotation to web-published contents so that web information can be effectively retrieved and processed by both humans and machines in a great variety of tasks. This is in practice achieved by using SW technologies, which enable to attach semantics to resources, ranging from very simple to very complex annotations depending on the requirements. SW technologies open a new dimension to data integration by providing a common terminology, standard for-

*Corresponding author
Email addresses: romerom@lsi.uji.es (Victoria Nebot), berlanga@lsi.uji.es (Rafael Berlanga)

mats for knowledge resources (e.g., RDF/(S)[1] and OWL[2]), semantically linked data, and more formal representations like description logic axioms for inference tasks. As a result, more and more semi-structured data and knowledge resources are being published in the Web, creating what is called the Web of Data [11]. At the same time, there is a great urgency of new analytical tools able to summarize and exploit all these resources.

Semantic search has emerged as one of the first applications making use and exploiting the Web of Data. New systems that offer search and browsing over RDF data have been developed [20, 19, 39, 16, 40]. Research in this area is a hot topic because it enhances conventional information retrieval by providing search services centered on entities, relations, and knowledge. Also, the development of the SW demands enhanced search paradigms in order to facilitate acquisition, processing, storage, and retrieval of semantic data. Albeit interesting, semantic search does not provide users with great insight into the data. Various advanced applications that have recently emerged impose modern user and business needs that require a more analytical view. Some examples include customer support, product and market research or life sciences and health care applications with both structured and unstructured information available.

The biomedical domain is one of the most active areas where significant efforts have been spent to export database semantics to data representation formats following open standards, which explicitly state the semantics of the content. The use of ontologies and languages such as RDF and OWL in the area of medical sciences has already a long way, and several web references to medical ontologies for different areas of medicine can be found (UMLS [12], SNOMED-CT [6], GALEN [1], GO [2], etc.) Moreover, a growing penetration of these technologies into the life science community is becoming evident through many initiatives (e.g. Bio2RDF [9], Linked Life Data [3], Liking Open Drug Data [4], etc.) and an increasing number of biological data providers, such as UniProt [7], have started to make their data available in the form of triples. In these complex scenarios keyword or faceted search may be useful but not enough, since advanced analysis and understanding of the information stored are required. Given the previous analysis requirements over these new complex and semantic representation formats, the well-known data warehousing (DW) and OLAP technologies seem good candidates to perform more insightful analytical tasks.

During the last decade, DW has proved its usefulness in traditional business analysis and decision making processes. A data warehouse can be described as a decision-support tool that collects its data from operational databases and various external sources, transforms them into information and makes that information available to decision makers in a consolidated and consistent manner [24]. One of the technologies usually associated to DW is multidimensional (MD) processing, also called OLAP [17] processing. MD models define the an-

[1] RDF/(S): `http://www.w3.org/TR/rdf-concepts/ , rdf-schema/`, '04
[2] OWL: `http://www.w3.org/TR/owl-features/`, '04.

alytic requirements for an application. It consists of the events of interest for an analyst (i.e., facts) described by a set of measures along with the different perspectives of analysis (dimensions). Every fact is described by a set of dimensions, which are organized into hierarchies, allowing the user to aggregate data at different levels of detail. MD modeling has gained widespread acceptance for analysis purposes due to its simplicity and effectiveness.

This work opens new interesting research opportunities for providing more comprehensive data analysis and discovery over semantic data. In this paper, we concentrate on semantic annotations that refer to an explicit conceptualization of entities in the respective domain. These relate the syntactic tokens to background knowledge represented in a model with formal semantics (i.e. an ontology). When we use the term "semantic", we thus have in mind a formal logical model to represent knowledge. We provide the means for efficiently analyzing and exploring large amounts of semantic data by combining the inference power from the annotation semantics with the analysis capabilities provided by OLAP-style aggregations, navigation, and reporting. We formally present how semantic data should be organized in a well-defined conceptual MD schema, so that sophisticated queries can be expressed and evaluated. As far as we know there is no tool providing OLAP-like functionality over semantic web data. In Section 2.3 we elaborate on the requirements and challenges faced when dealing with semantic web data.

The main contributions of the paper are summarized as follows:

- We introduce a semi-automatic method to dynamically build MD fact tables from semantic data guided by the user requirements.
- We propose two novel algorithms to dynamically create dimension hierarchies with "good" OLAP properties from the taxonomic relations of domain ontologies for the fact tables generated with the previous method.
- We provide an application scenario and a running use case in the biomedical domain to illustrate the usefulness and applicability of our method.

The rest of the paper is organized as follows. Section 2 introduces the application scenario with a use case that motivates our approach. Section 3 provides an overview of the method. Section 4 contains the user conceptual MD schema definition. Section 5 is devoted to the Fact Extraction process, and we cover both the foundations and implementation issues. In Section 6, we propose two methods for Dimension Extraction. Section 7 describes the experimental evaluation and implementation. Section 8 reviews related work regarding the trends in analytical tools for non-conventional data sources, mainly the semi-structured ones and finally, in Section 9 we give some conclusions and future work.

2. Application scenario

In this paper we adopt the application scenario of the Health-e-Child integrated project (HeC) [37], which aimed to provide a grid-based integrated data

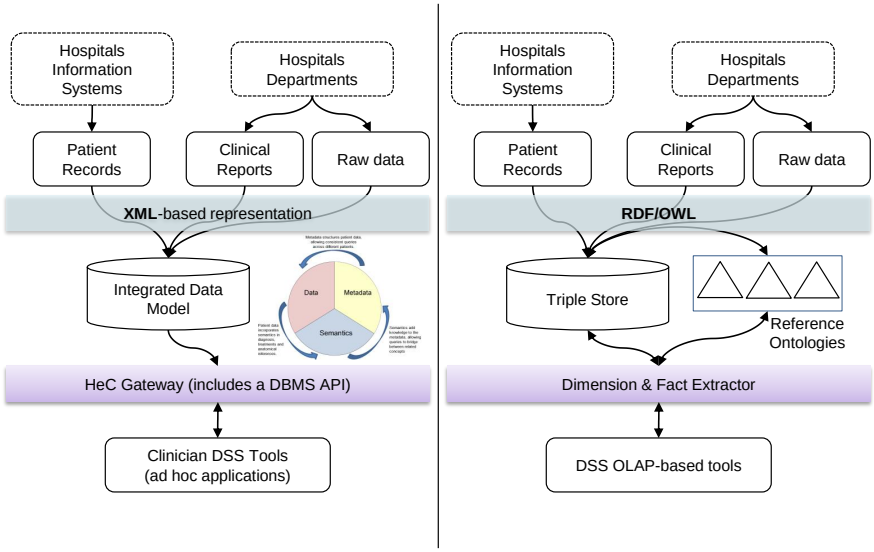

Figure 1: HeC data integration architecture (left hand) versus the Semantic Web integration architecture (right hand)

infrastructure for new decision support systems (DSS) for Paediatrics. Such DSS tools are mainly focused on traditional diagnosis/prognosis tasks and patient follow-up.

This scenario regards three main types of data sources: well standardized records coming from hospital information systems (e.g. HL7-conformant records), highly heterogeneous semi-structured clinical reports, and a variety of unstructured data such as DICOM files, ECG data, X-ray and ultrasonography images. All data are delivered to the HeC infrastructure in XML format, usually lacking of any schema for the semi- and unstructured data sources. Unfortunately, most of the interesting analysis dimensions are located in the schema-less data sources. The integration solution proposed in the HeC project formerly consisted of defining a flexible integrated data model [10], which is built on top of a grid-aware relational database management system (see left hand side of Figure 1). As clinical reports present very irregular structures (see the example presented in Figure 2), they are decomposed and classified into a few organizational units (i.e. patient, visit, medical event and clinical variables), thus disregarding much information that could be useful for the intended DSS tools. Additionally, this data model also includes tables that map clinical variables to concept identifiers from the UMLS Metathesaurus [12]. These tables are intended to link patient data to bibliographic resources such as MEDLINE [5]. It is worth mentioning that a similar scenario and solution have been proposed in

the OpenEHR initiative[3].

However, in this paper we adopt a different integration architecture (see right hand side of Figure 1), which follows the current trends in biomedical [13] and bioinformatics data integration [27], and relies entirely on SW technology. In this architecture, the integrated data model is defined as an *application ontology* that models the health care scenario (e.g. patients, visits, reports, etc.) This ontology can import concepts defined in external knowledge resources (*reference domain ontologies* in Figure 1). Finally, the biomedical data is semantically annotated according to the application ontology and stored as RDF triples (i.e. triple store). In the HeC scenario, the application ontology modeling the rheumatology domain has 33 concepts and 39 roles. However, this ontology can import any concept from UMLS, which is used as reference domain ontology, and contains over 1.5 million concepts. The triple store of the application scenario has 605420 instances containing references to 874 different concepts of UMLS. Moreover, depending on the user's requirements (i.e. dimensions selected) more domain ontology axioms will be imported in order to build the dimension hierarchies.

In this paper, we aim at analyzing and exploiting all this semantic data through OLAP-based tools. Subsequent sections present the representation formalism of the semantic data, some examples of OLAP-based analysis, and finally, the main issues involved in carrying out OLAP-based analysis over semantic data.

2.1. Semantic Data Representation

We use the language OWL-DL to represent the application ontology to which semantic annotations refer. OWL-DL has its foundation in the description logics (DL) [8]. Basically, DLs allow users to define the domain of interest in terms of *concepts* (called *classes* in OWL), *roles* (called *properties* in OWL) and *individuals* (called *instances* in OWL). Concepts can be defined in terms of other concepts and/or properties by using a series of constructors, namely: concept union ($\sqcup$), intersection ($\sqcap$) and complement ($\neg$), as well as enumerations (called *oneOf* in OWL) and existential ($\exists$), universal ($\forall$) and cardinality ($\geq$, $\leq$, $=$) restrictions over a role R or its inverse R^{-}.

Concept definitions are asserted as axioms, which can be of two types: concept subsumption ($C \sqsubseteq D$) and concept equivalence ($C \equiv D$). The former one is used when one only partially knows the semantics of a concept (the necessary but not sufficient conditions). Equivalence and subsumption can be also asserted between roles, and roles can have special constraints (e.g., transitivity, symmetry, functionality, etc.) The set of asserted axioms over concepts and roles is called the terminological box ($TBox$).

Semantic data is expressed in terms of individual assertions, which can be basically of two types: assert the concept C of an individual a, denoted $C(a)$,

[3] `http://www.openehr.org/home.html`

```
Ultrasonography
     WristExamination
        date '10/10/2006'
        hasUndergoneWrist True
        rightWrist False
        leftWrist True
     WristScore
        wristExamined 'Left'
        PannusAndJointEffusion
           distalRadioUlnarJoint
              result 'No Synovial thikening and no joint effusion'
           radioCarpalJoint
              result 'Only synovial pannus without joint efussion
           midCarpalCMCJ
              result 'None'
        Synovitis
           distalRadioUlnarJoint 'Mild'
           radioCarpalJoint 'None'
           midCarpalCMCJ 'Severe'
        BoneErosion
           distalUlna
              erosionDegree 0
           carpalBones
              erosionDegree 1
...
```

Figure 2: Fragment of a clinical report of the Rheumatology domain.

and assert a relation between two individuals a and b, denoted $R(a, b)$. The set of assertions over individuals is called the assertional box ($ABox$).

For practical purposes, the $TBox$ and the $ABox$ are treated separately. Notice that while the $ABox$ is usually very dynamic for it is constantly updated, the $TBox$ hardly changes over time. From now on, we will use the term *ontology* to refer to the $TBox$, and *instance store* to refer to the $ABox$ storage system. We assume that the instance store is always consistent w.r.t. the associated ontology.

Figure 3 shows a fragment of the ontology designed for patients with rheumatic diseases in our application scenario, whereas Table 1 shows a fragment of an *instance store* associated to this ontology. In this case, the instance store is expressed as triples $(subject, predicate, object)$, where a triple of the form $(a, type, C)$ corresponds to a DL assertion $C(a)$, and otherwise the triple (a, R, b) represents the relational assertion $R(a, b)$.

2.2. Example of Use Case

In the previous application scenario, we propose as use case to analyze the efficacy of different drugs in the treatment of inflammatory diseases, mainly rheumatic ones. At this stage, the analyst of this use case should express her analysis requirements at a conceptual level. Later on, these requirements will be translated into dimensions and measures of a MD schema that will hold semantic data.

Figure 4 depicts the conceptual model of the analyst requirements where the central subject of analysis is *Patient*. The analyst is interested in exploring the patient's follow-up visits according to different parameters such as gender,

$$
\begin{aligned}
Patient &\sqsubseteq\ =1\ hasAge.string\\
Patient &\sqsubseteq\ =1\ sex.Gender\\
Patient &\sqsubseteq \forall\ hasGeneReport.GeneProfile\\
GeneProfile &\sqsubseteq \forall\ over.Gene\ \sqcap\ \forall\ under.Gene\\
Patient &\sqsubseteq \forall\ hasHistory.PatientHistory\\
PatientHistory &\sqsubseteq \exists familyMember.Family\_Group\ \sqcap\ \exists\ hasDiagnosis.Disease\_or\_Syndrome\\
Patient &\sqsubseteq \exists\ hasVisit.Visit\\
Visit &\sqsubseteq\ =1\ date.string\\
Visit &\sqsubseteq \forall\ hasReport.(Rheumatology\ \sqcup\ Diagnosis\ \sqcup\ Treatment\ \sqcup\ Laboratory)\\
Rheumatology &\sqsubseteq \exists\ results.(Articular\ \sqcup\ ExtraArticular\ \sqcup\ Ultrasonography)\\
Rheumatology &\sqsubseteq\ =1\ damageIndex.string\\
Ultrasonography &\sqsubseteq \forall\ hasAbnormality.Disease\_or\_Syndrome\\
Ultrasonography &\sqsubseteq \forall\ location.Body\_Space\_or\_Junction\\
ArticularFinding &\sqsubseteq \exists\ affectedJoint.Body\_Space\_or\_Junction\\
ArticularFinding &\sqsubseteq \forall\ examObservation.string\\
Diagnosis &\sqsubseteq \exists\ hasDiagnosis.Disease\_or\_Syndrome\\
Treatment &\sqsubseteq\ =1\ duration.string\\
Treatment &\sqsubseteq \exists\ hasTherapy.DrugTherapy\\
DrugTherapy &\sqsubseteq\ =1\ administration.AD\\
DrugTherapy &\sqsubseteq\ =1\ hasDrug.Pharmacologic\_Substance\\
AD &\sqsubseteq \exists\ dosage.string\ \sqcap\ \exists\ route.string\ \sqcap\ \exists\ timing.string\\
Laboratory &\sqsubseteq \exists\ bloodIndicants.(\ \exists\ cell.Cell\ \sqcap\ \exists\ result.string\ \sqcap\ \exists\ test.Lab\_Procedure)\\
Rheumatoid\_Arthritis &\sqsubseteq Autoimmune\_Disease\\
Autoimmune\_Disease &\sqsubseteq Disease\_or\_Syndrome\\
&\ldots
\end{aligned}
$$

Figure 3: Ontology axioms (Tbox).

subject	**predicate**	**object**
PTNXZ1	hasAge	“10”
PTNXZ1	sex	Male
VISIT1	date	“06182008”
VISIT1	hasReport	RHEX1
RHEX1	damageIndex	“10”
RHEX1	results	ULTRA1
ULTRA1	hasAbnormality	“Malformation”
ULTRA1	hasAbnormality	Knee
VISIT1	hasReport	DIAG1
DIAG1	hasDiagnosis	Arthritis
VISIT1	hasReport	TREAT1
TREAT1	hasDrugTherapy	DT1
DT1	hasDrug	Methotrexate
PTNXZ1	hasVisit	VISIT2
VISIT2	date	“08202008”
VISIT2	hasReport	RHEX2
RHEX2	damageIndex	“15”
RHEX2	results	ULTRA2
ULTRA2	hasAbnormality	“Malformation”
ULTRA2	hasAbnormality	Knee
RHEX2	results	ULTRA3
ULTRA3	hasAbnormality	“Rotation 15degrees”
ULTRA3	hasAbnormality	Right_Wrist
VISIT2	hasReport	DIAG2
DIAG2	hasDiagnosis	Systemic_Arthritis
VISIT2	hasReport	TREAT2
TREAT2	hasDrugTherapy	DT2
DT2	hasDrug	Methotrexate
TREAT2	hasDrugTherapy	DT3
DT3	hasDrug	Corticosteroids
...	...	...

Table 1: Semantic annotations (Abox).

age, the drugs prescribed, the affected body parts, the diseases diagnosed, the articular damage and the number of occurrences.

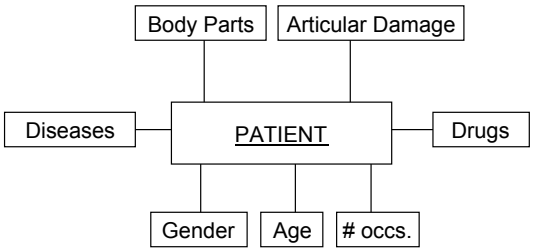

Figure 4: Use case analysis requirements.

Later on, the previous analysis requirements will be formally expressed in terms of DL expressions over the ontology and these expressions will be mapped to elements of a MD schema (i.e. dimensions and measures) that will eventually be populated with the semantic annotations of the instance store. The ultimate goal of the approach is to effectively analyze and explore the semantic annotations using OLAP-style capabilities. In this use case, the analyst will be able to specify useful MD queries in order to analyze patient data and discover useful patterns and trends. As an example, we show two MD queries expressed in MDX-like syntax that an analyst would execute over the resulting fact table in order to generate cubes. The first query builds a cube where for each administered drug and diagnosed disease the articular damage registered is averaged. By exploring this cube the analyst can find out interesting patterns such as which drugs mitigate the articular damage depending on the disease type.

```
CREATE CUBE [cubeArticularDamage1]
FROM [patient_dw]
(
   MEASURE [patient_dw].[avgArtDamIndex],
   DIMENSION [patient_dw].[Drug],
   DIMENSION [patient_dw].[Disease]
)
```

The second query builds a cube where the articular damage is explored over different analysis perspectives, namely the affected body parts and the patient's gender. This way the analyst can discover which body parts contribute most to the total damage index contrasted by gender.

```
CREATE CUBE [cubeArticularDamage2]
FROM [patient_dw]
(
   MEASURE [patient_dw].[avgArtDamIndex],
   DIMENSION [patient_dw].[BodyPart],
   DIMENSION [patient_dw].[Gender]
)
```

2.3. Issues in analyzing semantic data

The full exploitation of semantic data by OLAP tools is far from trivial due to the special features of semantic data and the requirements imposed by OLAP tools. Next, we describe some of the main challenges we encounter when dealing with semantic web data, which are treated in this paper:

- Usability: When dealing with the Web of Data, the user typically needs to specify a structured query in a formal language like SPARQL. However, the end user often does not know the query language and the underlying data graph structure. Even if she did, languages such as SPARQL do not account for the complexity and structural heterogeneity often common in RDF data. We overcome this issue by providing the user with a simple mechanism to specify her analysis requirements at the conceptual level. We ask the user to compose a conceptual MD schema by selecting concepts and properties of interest from the available ontologies.

- Imprecision: The information needs expressed by the user might be imprecise. On one hand, the flexibility and ease of use offered to the user in the requirements phase can lead to an ambiguous specification (i.e. the concepts and properties selected might be used in several contexts in the ontologies). On the other hand, the user might have limited knowledge about the domain and her specification might be too general or abstract. Our method overcomes both types of imprecision by taking into account all possible interpretations of the concepts specified by the user. Moreover, thanks to the semantics attached to the data, implicit information can be derived and made explicit so that the user can refine her MD specification by selecting these new inferred concepts.

- Scalability: As the amount of available data is ever growing, the ability to scale becomes essential. What is more important, scalability is known to be an issue when reasoning with large ontologies. In this work, we overcome this issue by applying specific indexes to ontologies in order to handle basic entailments minimizing the use of the reasoner.

- Dynamicity: The web is continuously changing and growing. Therefore, analytical tools should reflect these changes and present fresh results. Also, data are being provided at an overwhelming level of detail and only a subset of attributes constitute meaningful dimensions and facts for the analyst. For these reasons, our method allows the user to select only the concepts of interest for analysis and automatically derives a MD schema (i.e. dimensions and facts) and its instantiation ready to be fed into an off the shelf OLAP tool, for further analysis.

3. Method overview

In this section we present an overview of our method. We focus on the end-to-end description of the data flow from expressing the analysis requirements

to returning analytical query results. Figure 5 depicts a schematic overview of the whole process. The repository of semantic annotations on the left of Figure 5 stores both the application and domain ontology axioms (i.e. *Tbox*) and the semantic annotations (i.e. instance store). This repository maintains the necessary indexes for an efficient management. For obtaining OLAP style functionality, our goal is to create a MD schema from both the analyst requirements and the knowledge encoded in the ontologies. The fact table is populated with facts extracted from the semantic annotations –a fact is not a simple RDF triple but a valid semantic combination of structurally related instances– while dimension hierarchies are extracted by using the subsumption relationships inferred from the domain ontologies. Each of the phases are described in turn.

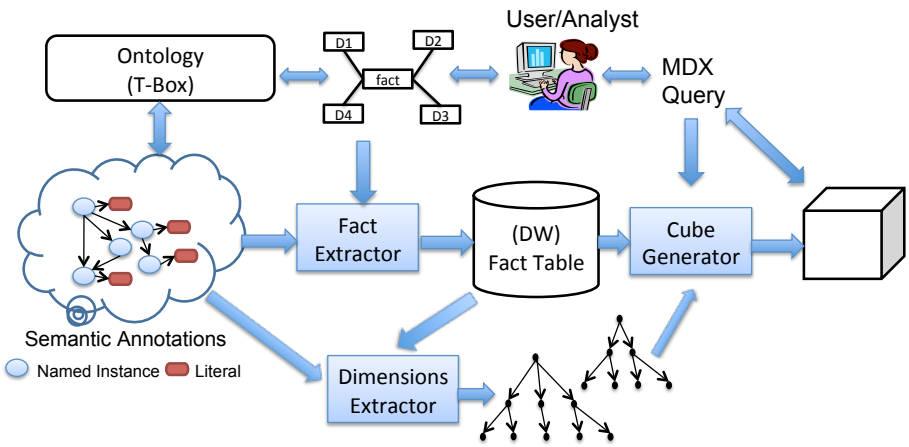

Figure 5: Architecture for semantic annotations analysis.

- During the design of the conceptual MD schema, the user expresses her analysis requirements in terms of DL expressions over the ontology. In particular, she selects the subject of analysis, the potential semantic types for the dimensions and the measures.
- The fact extractor is able to identify and extract facts (i.e. valid combinations of instances) from the instance store according to the conceptual MD schema previously designed, giving rise to the base fact table of the DW.
- The dimensions extractor is in charge of building the dimension hierarchies based on the instance values of the fact table and the knowledge available in the domain ontologies (i.e. inferred taxonomic relationships) while also considering desirable OLAP properties for the hierarchies.
- Finally, the user can specify MD queries over the DW in order to analyze and explore the semantic repository. The MD query is executed and a

cube is built. Then, typical OLAP operations (e.g., roll-up, drill-down, dice, slice, etc.) can be applied over the cube to aggregate and navigate data.

4. Multidimensional schema definition

In this section, we present how the analyst information requirements are translated into a conceptual MD schema in order to leverage the aggregation power of OLAP. The analyst defines the MD elements with DL expressions, which seems the most natural choice given that the data sources are expressed under this logical formalism. In particular, the MD schema contains the following elements:

- *Subject of analysis*: the user selects the subject of analysis from the ontology concepts. We name this concept C_{SUB}.
- *Dimensions*: usually, a dimension is defined as a set of levels with a partial order relation among them (i.e., hierarchy). We postpone the hierarchy creation of each dimension in order to dynamically adapt it to the base dimension values appearing in the resulting fact table. Therefore, the analyst must specify just the semantic type of each dimension. They can be either concepts, named C, or data type properties, named *dtp*, selected from the ontology.
- *Measures*: The user selects measures from the ontology by specifying the numeric data type property *dtp* and the aggregation function to be applied (i.e., sum, average, count, min, max, etc).

Notice that at this point, the analyst does neither know the dimension values nor the roll-up relationships that will eventually be used in the resulting MD schema. The method presented in this paper will automatically capture this information from the application and the domain ontologies involved in the analysis but always taking into account the user requirements. The syntax for the definition of the MD elements is specified in Figure 6 by means of a version of Extended BNF where terminals are quoted and non-terminals are bold and not quoted. For the running use case, the MD schema definition is shown in Figure 7.

5. Extracting Facts

This section addresses the process of fact extraction from the semantic annotations. First, the foundations of the approach are presented and then we discuss implementation details.

subject ::= $'SUBJECT('$ **name** $','$ **concept** $')'$
dimension ::= $'DIM('$ **name** $','$ **concept** | **datatypeprop**$')'$
measure ::= $'M('$ **name** $','$ **numValueFunction** $'('$**datatypeprop**$'))'$
name ::= **identifier**
concept ::= **concept definition according to OWL DL syntax**
datatypeprop ::= **datavaluedPropertyID**
numValueFunction ::= $'AVG'$ | $'SUM'$ | $'COUNT'$ | $'MAX'$ | $'MIN'$
datavaluedPropertyID ::= **URIreference**
identifier ::= **valid string identifier**

Figure 6: BNF syntax for MD schema specification

$SUBJECT(\ PatientCase\ ,\ Patient\ )$
$DIM(\ Disease\ ,\ Disease\_or\_Syndrome)$
$DIM(\ Drug\ ,\ Pharmacological\_Substance)$
$DIM(\ Age\ ,\ hasAge)$
$DIM(\ Gender\ ,\ Gender)$
$DIM(\ BodyPart\ ,Body\_Space\_Or\_Junction)$
$M(\ AvgArtDamIndex\ ,\ AVG\ (damageIndex)$
$M(\ NumberOccs.\ ,\ COUNT\ (PatientCase))$

Figure 7: MD schema specification for the use case

5.1. Foundations

The ultimate goal of the approach is to identify and extract valid facts according to the user's requirements. However, the user's conceptual MD schema is ambiguous in the sense that it can have several interpretations. In the running example, the dimension *Disease* can refer to the patient's main diagnosis, to some family member diagnosis, or it can be a collateral disease derived from the main disease detected through laboratory tests or rheumatic exams. In absence of further knowledge, the fact extraction process should account for all possible interpretations of the user's conceptual MD schema. Later on, when constructing OLAP cubes, the system will ask the user for her interpretation of the dimensions and measures in case there is more than one. This section elaborates on the definitions that allow to capture all possible interpretations of the user analysis requirements and thus extract all possible facts under these different interpretations. From now on, O refers to the ontology axioms, IS to the instance store and C_{SUB}, D and M are the subject of analysis, dimensions and measures, respectively. We show examples of the definitions with a simplified version of the user requirements for the sake of readability and comprehension. Suppose the analyst is interested in analyzing rheumatic patients with the following dimensions and measures: $C_{SUB} = Patient$, $D = \{Gender, Drug, Disease\}$, $M = \{damageIndex\}$. Figure 8 shows a graph-based representation of the ontology fragment where the schema elements are shaded.

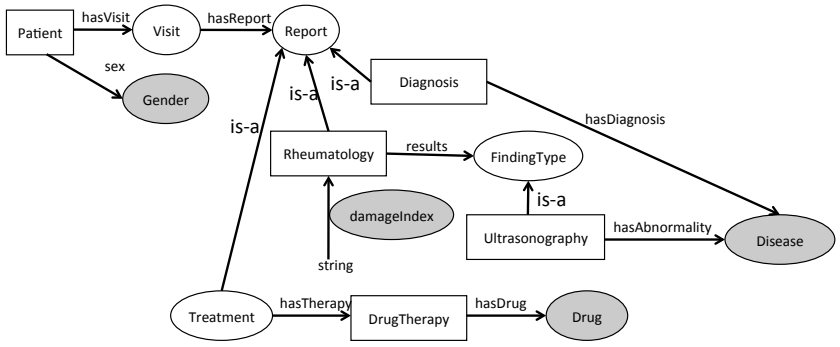

Figure 8: Example of ontology graph fragment.

Definition 1. *The subject instances are the set* $I_{SUB} = \{i/i \in IS,\ O \cup IS \models C_{SUB}(i)\}$.

In the running example C_{SUB} is $Patient$ and I_{SUB} is the set of all instances classified as $Patient$.

Definition 2. *Let* $c \in D$ *be a dimension. We define the senses of* c *as the following set:*

$$senses(c) = MSC(\{c' | \exists r \in O, O \models (c' \sqsubset \exists r.c)\} \cup \top)$$

where the function MSC[4] *returns the most specific concepts of a given concept set, that is, those concepts of the set that do not subsume any other concept in the same set.*

Similarly, we can define the senses of a measure $p \in M$ *as follows:*

$$senses(p) = MSC(\{c' | O \models c' \sqsubset \exists p.\top\} \cup \top)$$

Finally, we define S *as the union of the senses of each dimension and measure:*

$$S = \bigcup_{\forall i \in D \cup M} senses(i)$$

Example 1. *In Figure 8 the senses of each MD element are enclosed in boxes.* $senses(Drug) = \{DrugTherapy\}$, $senses(Disease) = \{Diagnosis, Ultrasonography\}$, $senses(Gender) = \{Patient\}$ *and* $senses(damageIndex) = \{Rheumatology\}$. *Notice that the senses are the different interpretations that the MD elements can have in the ontology.*

We need to identify the senses for both dimensions and measures because dimension values can participate in different parts of the ontology and measures can also be applied to different domain concepts.

The following definitions (3-6) are aimed at capturing the structural properties of the instance store w.r.t. the ontology in order to define valid instance combinations for the MD schema.

Definition 3. *The expression* $(r_1 \circ ... \circ r_n) \in Paths(C, C')$ *is an aggregation path from concept* C *to concept* C' *of an ontology* O *iff* $O \models C \sqsubseteq \exists\, r_1 \circ ... \circ r_n.C'$.

Definition 4. *Let* $C_a, C_b \in S$ *be two named concepts.* $Contexts(C_a, C_b, C_{SUB}) = \bigcup_{\forall C' \in LCRC(C_a, C_b, C_{SUB})} \{C''/C'' \sqsubseteq C'\}$, *where the function* $LCRC(C_a, C_b, C_{SUB})$ *returns the set of least common reachable concepts,*

1. $|Paths(C', C_a)| > 0 \wedge |Paths(C', C_b)| > 0 \wedge |Paths(C_{SUB}, C')| > 0$ *(*C' *is common reachable concept).*
2. $\nexists E \in O$ *such that* E *satisfies the condition 1 and* $|Paths(C', E)| > 0$ *(*C' *is least).*

The first condition states that there must be a path connecting C' and C_a, another path connecting C' and C_b and a third path connecting the subject concept C_{SUB} with C'. The previous definition is applied over all the senses in S pairwise in order to find common contexts that semantically relate them.

[4]Do not confuse with the inference task in DL with the same name, MSC of an instance.

Example 2. *In Figure 8,* $Contexts(DrugTherapy, Diagnosis, Patient) = \{Visit\}$ *because* $p_1 = Visit.hasReport \circ hasTherapy.DrugTherapy$, $p_2 = Visit.hasReport.Diagnosis$ *and* $p_3 = Patient.hasVisit.Visit$.

Definition 5. *Let* $i, i' \in IS$ *be two named instances. The expression* $(r_1 \circ \cdots \circ r_n) \in Paths(i, i')$ *is an aggregation path from* i *to* i' *iff:*

1. *There exists a list of property assertions* $r_j(i_{j-1}, i_j) \in IS$, $1 \leq j \leq n$.
2. *There exists two concepts* $C, C' \in O$ *such that* $O \cup IS \models C(i) \wedge C'(i')$ *and* $(r_1 \circ \cdots \circ r_n) \in Paths(C, C')$

The first condition of the previous definition states that there must be a property chain between both instances in the *IS*, and the second one states that such a path must be also derived from the ontology.

Definition 6. *Let* $i_a, i_b \in IS$ *be two named instances such that* $O \cup IS \models C_a(i_a) \wedge C_b(i_b) \ \wedge \ C_a, C_b \in S \ \wedge \ C_a \neq C_b$. $Contexts(i_a, i_b, i_{SUB}) = \bigcup_{\forall i' \in LCRI(i_a, i_b, i_{SUB})} \{i'\}$, *where the function* $LCRI(i_a, i_b, i_{SUB})$ *returns the set of least common reachable instances,*

1. $|Paths(i', i_a)| > 0 \ \wedge \ |Paths(i', i_b)| > 0 \ \wedge \ |Paths(i_{SUB}, i')| > 0$ *(*i' *is common reachable instance).*
2. $\nexists j \in IS$ *such that* j *satisfies the condition 1 and* $|Paths(i', j)| > 0$ *(*i' *is least).*

The previous definition is applied over instances belonging to different senses pairwise in order to find common contexts that relate them.

Example 3. *Figure 9 shows an example of IS represented as a graph that is consistent with ontology fragment in Figure 8. In this example,* $Contexts(DT1, DIAG1, PTN\_XY21) = \{VISIT1\}$ *because* $p_1 = VISIT1.hasReport \circ hasTherapy.DT1$, $p_2 = VISIT1.hasReport.DIAG1$ *and* $p_3 = PTN\_XY21.hasVisit.VISIT1$.

Now, we can define which instances can be combined to each other for a certain analysis subject C_{SUB}.

Definition 7. *Two instances* $i_a, i_b \in IS$ *are* combinable *under a subject instance* i_{SUB} *iff* $\exists\, C_1 \in \{C(i_x)/i_x \in Contexts(i_a, i_b, i_{SUB})\}, \exists\, C_2 \in Contexts(C(i_a), C(i_b), C_{SUB})$ *such that* $O \models C_1 \sqsupseteq C_2$ *where* $C(i)$ *is the asserted class for the instance* i *in* IS.

Example 4. *As an example of the previous definition, let us check if instances* $DT1$ *and* $DIAG1$ *of Figure 9 are combinable. In the definition,* C_1 *can only be* $Visit$ *and* C_2 *is also* $Visit$ *by looking at Figure 8. Since* $O \models Visit \sqsupseteq Visit$, *instances* $DT1$ *and* $DIAG1$ *are combinable under subject instance* PTN_XY21. *The intuition of the previous definition is that two instances are combinable only*

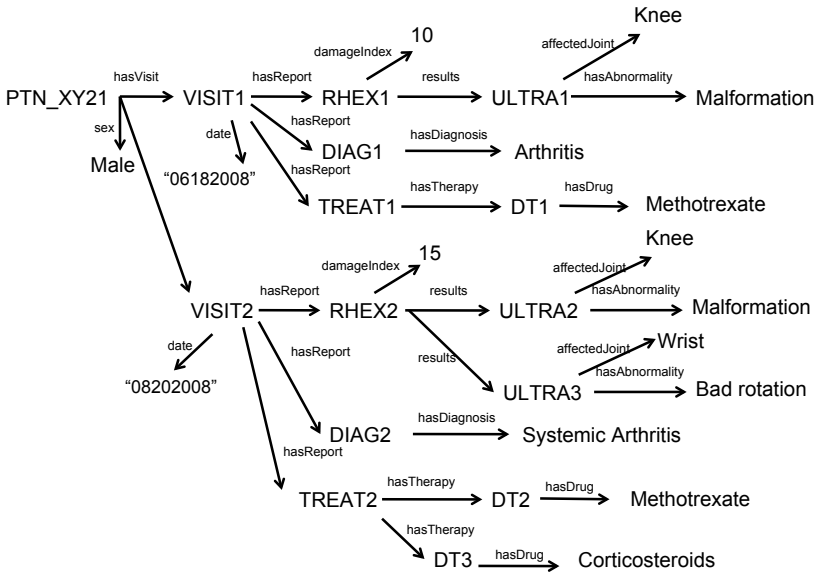

Figure 9: Example of instance store fragment consistent with ontology fragment of Figure 8.

if at the instance level (i.e., in the instance store or Abox*) they appear under a context that belongs to or is more general than the context specified at the conceptual level (i.e., in the* Tbox*). Following this intuition, instances* $DT1$ *and* $DIAG2$ *are not combinable because their contexts at the instance level are* $\{PTN\_XY21\}$ *whose type is Patient while at the conceptual level their contexts are* $\{Visit\}$*. This makes complete sense because the ontology has been defined in such a way that each visit contains the diseases diagnosed along with the drugs prescribed.*

Only valid combinations conformant with the MD schema are taken into account in the following definition.

Definition 8. *An instance context associated to* $i_{SUB} \in IS$ *is the tuple* $(i_1, \cdots, i_n)$*, with* $n \geq |D \cup M|$*, satisfying the following conditions:*

1. $\forall c \in \{D \cup M\}$*, there is at least one tuple element* i_x*,* $1 \leq x \leq n$*, such that* $O \cup IS \models C(i_x)$ *and* $C \in senses(c)$ *or* $i_x = NULL$ *otherwise. From now on, we will denote with* $dim(i_x)$ *to the MD element associated to* i_x *(i.e. c).*
2. $\forall (j,k)$*,* $1 \leq j,k \leq n$*,* $j \neq k$*,* (i_j, i_k) *are combinable instances under* i_{SUB}*.*

In other words, each of the elements of an instance context tuple is an instance that belongs to one sense and satisfies the combinable condition of Definition 7 with the rest of instances in the tuple. Therefore, an instance

Dimensions	**Disease**		**Drug**	**Gender**	**damageIndex**
Senses	**Ultrasono.**	**Diagnosis**	**DrugTherapy**	**Patient**	**Rheuma.**
PTN_XY21	ULTRA1	DIAG1	DT1	PTN_XY21	RHEX1
PTN_XY21	ULTRA2	DIAG2	DT2	PTN_XY21	RHEX2
PTN_XY21	ULTRA2	DIAG2	DT3	PTN_XY21	RHEX2
PTN_XY21	ULTRA3	DIAG2	DT2	PTN_XY21	RHEX2
PTN_XY21	ULTRA3	DIAG2	DT3	PTN_XY21	RHEX2

Table 2: Instance context tuples generated for the running example. The second row accounts for the senses associated to each dimension.

context tuple represents all the potential valid interpretations from IS for a MD schema specification since all the elements of the MD schema must be covered by at least one instance or the $NULL$ value if there is not such instance (i.e. in case of optional values in the ontology).

We say that an instance context tuple is *ambiguous* if there is more than one sense associated to the same dimension (measure). For example, tuples in Table 2 are ambiguous because the dimension $Disease$ has associated two senses in all the tuples.

Example 5. *Given dimensions $D = \{Disease, Drug, Gender\}$ and measures $M = \{damageIndex\}$ with their associated senses, Table 2 shows the instance context tuples generated for subject instance PTN_XY21. Notice we keep track of the subject of analysis (i.e. Patient) in a new column.*

Finally, we must project the intended values to obtain the facts that will populate the MD schema.

Definition 9. *A data fact associated to an instance context tuple $(i_1, \cdots, i_m)$, is a tuple $(d_1, \cdots, d_n)$ with $n \geq m$ such that*

$$d_k = \begin{cases} j_k & \text{if } dim(i_k) \in D \text{ and } j_k \in \prod_p(i_k) \text{ and } O \cup IS \models C(j_k) \text{ and } \\ & C \sqsubseteq dim(i_k) \\ v_k & \text{if } dim(i_k) \in M \text{ and } v_k \in \prod_{dim(i_k)}(i_k) \\ null & \text{otherwise} \end{cases}$$

where $\prod_p$ is the projection operation over an instance through the property p.

Notice that we take into consideration that the projection can be multivalued (i.e. more than one value can be accessed with the same property in an instance).

It is worth mentioning that "disambiguation" of instance tuples and data facts must be performed before building the OLAP cubes. Such a disambiguation can only be manually performed by the user according to her analysis requirements and the semantics involved. We will not treat further this issue as we consider it out of the scope of this paper.

Dimensions	**Disease**		**Drug**	**Gender**	**damageI.**
Senses	**Ultrasono.**	**Diagnosis**	**DrugTherapy**	**Patient**	**Rheuma.**
PTN_XY21	Malformation	Arthritis	Methotrexate	Male	10
PTN_XY21	Malformation	Systemic_Arthritis	Methotrexate	Male	15
PTN_XY21	Malformation	Systemic_Arthritis	Corticosteroids	Male	15
PTN_XY21	Bad rotation	Systemic_Arthritis	Methotrexate	Male	15
PTN_XY21	Bad rotation	Systemic_Arthritis	Corticosteroids	Male	15

Table 3: Data fact tuples generated for the running example.

Example 6. *The previous instance context tuples in Table 2 give rise to the data fact tuples in Table 3 where each instance has been projected according to the previous definition. In the data fact tuples, measures are treated the same way as dimensions. Only when data fact tuples are going to be loaded into a cube, measure values sharing dimension values are aggregated according to the aggregation function selected.*

Complexity Issues. In our method, the extraction of facts from the IS (Definition 9) requires the generation of different subsets of O (i.e. *senses*, *paths* and *contexts*), which in turn requires the use of a reasoner to perform inferences like $O \models \alpha$ and $O \cup IS \models \alpha$. The complexity of current reasoners depends on the expressiveness of both O and α, and consequently it directly affects to the efficiency of our method. It is worth mentioning that the expressivity of OWL DL leads to exponential complexity for these inferences. Fortunately, recently proposed OWL2 profiles (EL, QL and RL) have been demonstrated to be tractable for the main reasoning tasks (i.e. ontology consistency, class subsumption checking and instance checking). Additionally, definitions 4 to 8 require intensive processing over the inferred taxonomy of O and the underlying graph structure of IS. For this reason, we have designed a series of indexes and data structures aimed to efficiently perform the required operations. Next section is devoted to this issue.

5.2. Implementation

The process of identification and extraction of valid facts from a semantic data repository involves several steps. In the previous section we have explained the identification of facts by means of a conceptual MD schema defined by the user. We have also lied down the foundations of valid facts from a semantic point of view. In this section we present an implementation for fact extraction that resembles ETL processes.

Figure 10 sketches the steps involved in the fact extraction process. In the upper part of the figure we can see how ontology axioms (i.e., *Tbox*) are indexed in order to handle basic entailments and hence minimizing the need of a reasoner. This task needs to be performed just once for each ontology and it can be done off-line. The on-line phase is shown in the bottom part of Figure 10. The first task consists of creating the composition triples from the instance store (i.e., *Abox*) according to the user MD schema. These triples allow efficient

instance retrieval as well as reachability queries. Then, we generate *instance context* tuples (see Definition 8) from the composition triples with the help of the indexes. Since each context tuple can give rise to multiple facts, we project them over the desired attributes specified in the user MD schema, obtaining the intended *data fact* tuples (see Definition 9). Finally, we apply the required transformations over data fact tuples (e.g., normalizing or making partitions of numeric values). Next sections illustrate the index structures and their role in the fact extraction process.

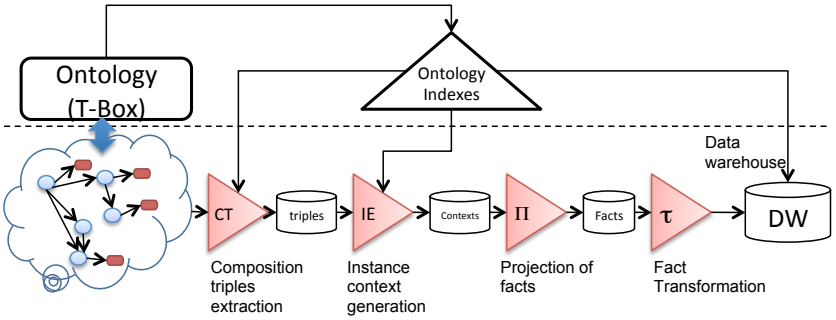

Figure 10: ETL process for fact extraction.

5.2.1. Ontology indexes

We have designed two indexes over the ontology in order to handle basic entailments namely: *is-a index* and *aggregation index*.

The is-a index is intended to efficiently answer concept subsumption queries and ancestors/descendants range queries. Briefly, the construction of the index involves the following steps: the *TBox* is normalized and completion rules are applied to make implicit subsumptions explicit. This process is similar to that presented in [14]. Then, all "is-a" relationships are modeled as a graph where nodes correspond to concepts and edges to "is-a" relationships. An interval labeling scheme able to encode transitive relations is applied to the graph. As a result, each node (i.e., concept) has assigned a descriptor that includes compressed information about its descendant and ancestor nodes in the form of intervals. Moreover, we provide an interval's algebra over the node descriptors that allows performing basic DL operations over concepts. A detailed description of the indexing process and interval's algebra can be found in [30].

The *aggregation index* is an index structure that allows answering reachability queries between concepts (see Definition 3). In order to construct this index, we apply the same interval labeling scheme as before over the ontology graph $G = (V, L, E)$, where nodes in V are associated to the ontology classes, edge labels L represent the object property relation names and the "is-a" relation and edges $e(v_1, v_2) \in E$ represent the asserted relationships $e \in L$ between nodes $v_1, v_2 \in V$. By encoding descendants (i.e., reachable nodes following edge

direction) and ancestors (i.e., reachable nodes following inverse edge direction) for each node, we are able to encapsulate reachable nodes through both explicit and implicit aggregation paths. Going back to the graph fragment G showed in Figure 8, concept *Drug* has among its ancestors (i.e., reachable nodes) not only *DrugTherapy* and *Treatment*, but also *Visit* through inheritance of *hasReport* property of *Visit*. The operation $Contexts(C_a, C_b, C_{SUB})$ (see Definition 4) can be easily implemented using the aggregation index as follows:

$$Contexts(C_a, C_b, C_{SUB}) = \bigcup_{C \in nca(C_a, C_b) \cap descendants(C_{SUB})} (ancestors(C) \cup C)$$

where operations *nca*, *descendants* and *ancestors* are efficiently performed through the interval's algebra previously mentioned.

5.2.2. Instance store index

The main purpose of the instance store (IS) index is to ease the task of finding out if two instances, or an instance and a literal, are connected in the instance store (see Definition 5). This index is materialized as composition triples.

Definition 10. *A* composition triple *is a statement of the form* (s, o, p) *where the subject (s) and object (o) are resources, and the predicate (p) is a composition path from the subject to the object. A composition path is an alternate sequence of properties and resources that connect both the subject and the object.*

For our purposes, the instance store index only keeps the composition triples that go from each instance of C_{SUB} (i.e., the subject of analysis) to each of the instances i such that $C(i)$ where C is a *sense* of the dimensions and measures. Moreover, both the subject and object of a composition triple contain a reference to their concept type C in the *is-a index*. Consequently, we can perform searches over the composition triples not only at the instance level (i.e., by exact matching of instance names in the subject and object) but also at the conceptual level (i.e., by specifying the subject and object concepts). The main operation defined over the composition triples is $get\_instances(subject, path, object)$, where *subject* and *object* can be either a concept or instance, and *path* is a pattern matching string over the intended paths.

Notice that the number of composition triples between two instances is equal to the number of unique paths in the IS graph for these instances. If the graph contains cycles, this number can be infinite. To avoid this problem, we assume that the IS forms a DAG, which indeed usually occurs in real world applications. We also require the composition path between a subject and an object to be unique. This way the number of composition triples is limited to $|subjects| \times |objects|$ and the relation between a subject and an object is unambiguous, avoiding possible summarizability issues in the resulting facts due to redundancy.

Subject	Composition Path	Object	Type
PTN_XY21	/hasVisit	VISIT1	Visit
PTN_XY21	/hasVisit	VISIT2	Visit
PTN_XY21	/hasVisit/VISIT1/hasReport	RHEX1	Rheumatology
PTN_XY21	/hasVisit/VISIT1/hasReport/RHEX1/results	ULTRA1	Ultrasono.
PTN_XY21	/hasVisit/VISIT1/hasReport	DIAG1	Diagnosis
PTN_XY21	/hasVisit/VISIT1/hasReport/TREAT1/hasTherapy	DT1	DrugTherapy
PTN_XY21	/hasVisit/VISIT2/hasReport	RHEX2	Rheumatology
PTN_XY21	/hasVisit/VISIT2/hasReport/RHEX2/results	ULTRA2	Ultrasono.
PTN_XY21	/hasVisit/VISIT2/hasReport/RHEX2/results	ULTRA3	Ultrasono.
PTN_XY21	/hasVisit/VISIT2/hasReport	DIAG2	Diagnosis
PTN_XY21	/hasVisit/VISIT2/hasReport/TREAT2/hasTherapy	DT2	DrugTherapy
PTN_XY21	/hasVisit/VISIT2/hasReport/TREAT2/hasTherapy	DT3	DrugTherapy
...	...	...	...

Table 4: An excerpt of the composition triples of instance store fragment in Figure 9.

Table 4 shows an excerpt of composition triples for the IS fragment in Figure 9. The fourth column of the table shows the concept reference for the object. The concept reference for the subject is omitted because it is *Patient* for all the composition triples showed.

5.2.3. Instance Context & Data Fact Generation

Our method to generate instance context tuples according to Definition 8, does not perform an exhaustive search over the instance store, since checking the combinable condition (see Definition 7) for each pair of instances would result in a combinatorial explosion. Instead, we use the ontology axioms to just select proper contexts that lead to valid combinations of instances. For this purpose we define the following data structure.

Definition 11. *Let O be an ontology (only the Tbox), C_{SUB} the subject concept, $DM = D \cup M$ the set of dimensions and measures and S the set of senses of the dimensions and measures. The* Contexts Graph *(*CG*) is a graph-like structure generated from O that satisfies the following conditions:*

1. *The root of the CG is C_{SUB}.*
2. *The rest of the nodes of the CG are $DM \cup S \cup \{nd/\ \forall C_i, C_j, 1 \leq i, j \leq |S|, C_i, C_j \in S, nd \in contexts(C_i, C_j, C_{SUB})\}$.*
3. *There is one edge from node nd_i to node nd_j iff $|Paths(nd_i, nd_j)| > 0$, $\nexists\, nd_k \in nodes(CG)$ such that $|Paths(nd_i, nd_k)| > 0 \ \wedge\ |Paths(nd_k, nd_j)| > 0$, where nodes(CG) denotes the set of nodes of CG.*

The generation of the CG is performed by calculating context nodes of the dimensions and measures senses. This process is supported by the operations provided by the *aggregation index*.

Example 7. *Given the conceptual MD schema of Example 5, the CG according to the previous definition is shown in the upper part of Figure 11, where shaded nodes act as contexts. The actual dimensions and measures are obviated for the sake of readability.*

Algorithm 1 generates instance context tuples and uses the CG as a guide to process each subject instance (see Figure 11). The main idea is to process the CG in depth in order to recursively collect and combine instances under context node instances. They are collected with the operation $gI(i_{SUB}, path, CG\_node)$, where each output instance is a tuple and *path* is recursively constructed as the CG is processed, reflecting the actual path. Tuples are combined with the cartesian product until complete context tuples are obtained. The resulting instance context tuples are shown in Table 2. Data fact tuples are generated according to Definition 9 by projecting instances over the dimensions and measures. The resulting data fact tuples are shown in Table 3.

Algorithm 1 Fact_Extraction_Algorithm

Require: CG: contexts graph (as global var.),
CT: composition triples (as global var.),
$i_{SUB} =$ " $*$ ": subject instance,
$cpath =$ " $*$ ": composition path,
$CG\_node = CG.root$: node from CG
Ensure: ICT: valid instance context tuples (as global var.)
1: $tuples = \emptyset$
2: $Dvals = \emptyset$
3: $Cvals = \emptyset$
4: $instances = CT.get\_instances(i_{SUB}, cpath, CG\_node)$
5: **for** $i \in instances$ **do**
6: **if** $i_{SUB} ==$ " $*$ " **then**
7: $i_{SUB} = i$
8: **else**
9: $cpath = concatenate(cpath, i)$
10: **end if**
11: **for** $n \in CG.successors(CG\_node)$ **do**
12: **if** *not* $CG.context\_node(n)$ **then**
13: $v = CT.get\_instances(i_{SUB}, cpath, n)$
14: $add(Dvals, v)$
15: **end if**
16: **end for**
17: $Dvals = cartesian\_product(Dvals)$
18: **for** $n \in CG.successors(CG\_node)$ **do**
19: **if** $CG.context\_node(n)$ **then**
20: $L = Fact\_Extraction\_Algorithm(i_{SUB}, cpath, n)$
21: $add(Cvals, L)$
22: **end if**
23: **end for**
24: $Cvals = combine(Cvals)$
25: $combs = cartesian\_product(Dvals, Cvals)$
26: **if** $CG.is\_root\_node(CG\_node)$ **then**
27: $add(ICT, combs)$
28: **else**
29: $add(tuples, combs)$
30: **end if**
31: **end for**
32: **return** tuples

6. Extracting Dimensions

Once instance facts are extracted, the system can proceed to generate hierarchical dimensions from the concept subsumption relationships inferred from the ontology. These dimensions, however, must have an appropriate shape to

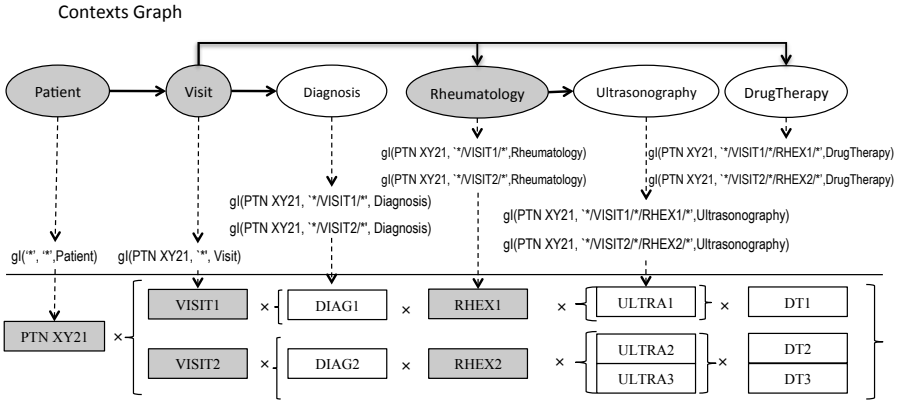

Figure 11: Contexts graph processing along with the generation of context tuples for instance store of Figure 9. Shaded nodes correspond to contexts. Function gI is the short name for function *get_instances*.

perform properly OLAP operations. Therefore extracted hierarchies must comply with a series of constraints to ensure summarizability [22]. In this section, we propose a method to extract "good" hierarchies for OLAP operations from the ontologies in the repository, but preserving as much as possible the original semantics of the involved concepts. In this work we only consider taxonomic relationships, leaving as future work other kind of relationships (e.g. transitive properties, property compositions, etc.), which have been previously treated in the literature [34].

6.1. Dimension Modules

The first step to extract a hierarchical dimension D_i consists of selecting the part of the ontology that can be involved in it. Let $Sig(D_i)$ be the set of most specific concepts of the instances participating in the extracted facts (see Definition 9).

We define the dimension module $M_{D_i} \subseteq O$ as the upper module of the ontology O for the signature $Sig(D_i)$. We define upper modules in the same way as in [21, 23, 30], that is, by applying the notion of conservative extension. Thus, an upper module M of O for the signature Sig is a sub-ontology of O such that it preserves all the entailments over the symbols of Sig expressed in a language $\mathcal{L}$, that is, $O \models \alpha$ with $Sig(\alpha) \subseteq Sig$ and $\alpha \in \mathcal{L}$ iff $M_{D_i} \models \alpha$.

For OLAP hierarchical dimensions we only need to preserve entailments of the form $C \sqsubseteq D$ and C *disjoint* D, being C and D named concepts. Thus, the language of entailments corresponds to the typical reasoners output. This kind of upper modules can be extracted very efficiently over very large ontologies [30].

6.2. Dimension Taxonomies

Let $TAX(M_{D_i})$ be the inferred taxonomy for the module M_{D_i}, which is represented as the directed acyclic graph (DAG) (V, E), where V contains one node for each concept in M_{D_i}, and $c_i \rightarrow c_j \in E$ if c_i is one of the least common subsumers of c_j (i.e. direct ancestor). $TAX(M_{D_i})$ is usually an irregular, unbalanced and non-onto hierarchy, which makes it not suitable for OLAP operations. Therefore, it is necessary to transform it to a more regular, balanced and tree-shaped structure. However, this transformation is also required to preserve as much as possible the original semantics of the concepts as well as to minimize the loss of information (e.g. under-classified concepts). Former work about transforming OLAP hierarchies [33] proposed the inclusion of fake nodes and roll-up relationships to avoid incomplete levels and double counting issues. Normalization is also proposed as a way to solve non-onto hierarchies [26]. These strategies however are not directly applicable to ontology taxonomies for two reasons: the number of added elements (e.g. fake nodes or intermediate normalized tables) can overwhelm the size of the original taxonomy and, the semantics of concepts can be altered by these new elements. Recently, [15] proposes a clustering-based approach to reduce taxonomies for improving data tables summaries. This method transforms the original taxonomy to a new one by grouping those nodes with similar structures and usage in the tuples. However, this method also alters the semantics of the symbols as new ones are created by grouping original ones.

6.3. Selecting good nodes

Similarly to our previous work about fragment extraction [30], we propose to build tailored ontology fragments that both preserve as much as possible the original semantics of the symbols in $Sig(D_i)$ and present a good OLAP-like structure. For the first goal, we will use the upper modules M_{D_i} as the baseline for the hierarchy extraction. For the second goal, we propose a series of measures to decide which nodes deserve to participate in the final hierarchy.

Before presenting the method, let us analyze why $TAX(M_{D_i})$ presents such an irregular structure. The usage of symbols in $Sig(D_i)$ can be very irregular due to the "popularity" of some symbols (i.e. Zipf law), which implies that few symbols are used very frequently whereas most of them are used few times. As a result, some parts of the taxonomy are more used than others, affecting to both the density (few dense parts and many sparse parts) and the depth of the taxonomy (few deep parts). A direct consequence is that some concepts in $Sig(D_i)$ are covered by many spurious concepts which are only used once, and therefore are useless for aggregation purposes. So, our main goals should be to identify dense regions of the taxonomy and to select nodes that best classify the concepts in $Sig(D_i)$.

The first measure we propose to rank the concepts in $TAX(M_{D_i})$ is the *share*:

$$share(n) = \prod_{n_i \in ancs(n)} \frac{1}{|children(n_i)|}$$

where $ancs(n)$ is the set of ancestors of n in $TAX(M_{D_i})$, and $children(n)$ is the set of direct successors of n in the taxonomy.

The idea behind the *share* is to measure the number of partitions produced from the root till the node n. The smaller the share the more dense is the hierarchy above the node. In a regular balanced taxonomy the ideal share is $S(n)^{depth(n)}$, where S is the mean of children the ancestor nodes of n have. We can then estimate the ratio between the ideal share and the actual one as follows:

$$ratio(n) = \frac{S(n)^{depth(n)}}{share(n)}$$

Thus, the greater the ratio, the better the hierarchy above the node is.

The second ranking measure we propose is the *entropy*, which is defined as follows:

$$entropy(n) = \Sigma_{n_i \in children(n)} P_{sig}(n, n_i) \cdot log(P_{sig}(n, n_i))$$

$$P_{sig}(n, n_i) = \frac{coveredSig(n_i)}{coveredSig(n)}$$

where $coveredSig(n)$ is the subset of $Sig(D_i)$ whose members are descendants of n.

The idea behind the entropy is that good classification nodes are those that better distribute the signature symbols among its children. Similarly to decision trees and clustering quality measures, we use the entropy of the groups derived from a node as the measure of their quality.

In order to combine both measures we just take the product of both measures:

$$score(n) = entropy(n) \cdot ratio(n)$$

6.4. Generating hierarchies

The basic method to generate hierarchies consists of selecting a set of "good" nodes from the taxonomy, and then re-construct the hierarchy by applying the transitivity property of the subsumption relationship between concepts. Algorithm 2 presents a *global* approach, which consists of selecting nodes from the nodes ranking until either all signature concepts are covered or there are no more concepts with a score greater than a given threshold (usually zero). Alternatively, we propose a second approach in Algorithm 3, the *local* approach, which selects the best ancestors of each concept leaf of the taxonomy. In both approaches, the final hierarchy is obtained by extracting the *spanning tree* that maximizes the number of ancestors of each node from the resulting reduced taxonomy [30]. One advantage of the local approach is that we can further select the number of levels up to each signature concept, defining so the hierarchical categories for the dimensions. However, this process is not trivial and we decided to leave it for future work. Each method favors different properties of the

generated hierarchy. If the user wants to obtain a rich view of the hierarchy, she must select the global one. Instead, if the user wants a more compact hierarchy (e.g., few levels) then she must select the local one.

Algorithm 2 Global approach for dimension hierarchy generation

Require: the upper module (M_{D_i}) and the signature set ($Sig(D_i)$) for dimension D_i.
Ensure: A hierarchy for dimension D_i.
  Let L_{rank} be the list of concepts in M_{D_i} ordered by $score(n)$ (highest to lowest);
  Let $Fragment = \emptyset$ be the nodes set of the fragment to be built.
  repeat
    pop a node n from L_{rank}
    add it to $Fragment$
  until $score(n) \leq 0$ or $\bigcup_{n \in Fragment} coveredSig(n) == Sig(D_i)$
  Let $NewTax$ be the reconstructed taxonomy for the signature $Fragment$
  return $spanningTree(NewTax)$

Algorithm 3 Local approach for dimension hierarchy generation

Require: the upper module (M_{D_i}) and the signature set ($Sig(D_i)$) for dimension D_i.
Ensure: A hierarchy for dimension D_i.
  Let L_{leaves} be the list of leaf concepts in M_{D_i} ordered by $ratio(n)$ (highest to lowest);
  Let $Fragment = \emptyset$ be the nodes set of the fragment to be built.
  for all $c \in L_{leaves}$ **do**
    Set up $L_{ancs}(c)$ with the ancestors of c ordered by $score(n)$
    for all $n_a \in L_{ancs}(c)$ **do**
      if there is no node $n_2 \in L_{ancs}(c)$ such that $score(n_2) \leq score(n_a)$ and $order(n_2) \leq order(n_a)$ **then**
        if $n_a \notin Fragment$ **then**
          add n_a to $Fragment$
        end if
      end if
    end for
  end for
  Let $NewTax$ be the reconstructed taxonomy for the signature $Fragment$
  return $spanningTree(NewTax)$

7. Evaluation

We have conducted the experiments by applying our method to a corpus of semantic annotations about rheumatic patients. The experimental evaluation has two differentiated setups. On one hand, we are concerned with scalability and performance issues regarding the generation of facts from the analyst requirements. On the other hand, we evaluate the two proposed methods for generating dimensions from the ontology knowledge by measuring the quality of the resulting hierarchies.

7.1. Fact extraction

The dataset used for the fact extraction evaluation has been synthetically generated from the features identified from a set of real patients. For this purpose, we have extended the XML generator presented in [36] with new operators

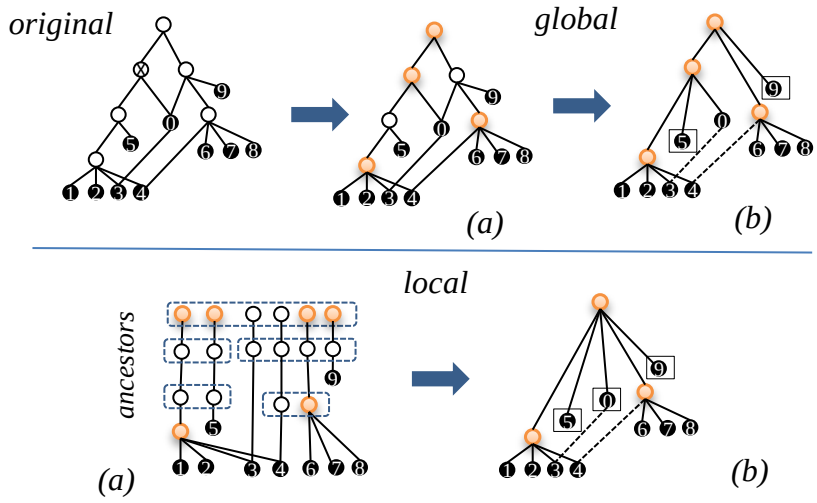

Figure 12: Example of local and global methods: (a) node selection and (b) hierarchy reconstruction. Dashed edges in (b) are removed in the final spanning tree. Nodes inside squares in (b) are those that change its parent in the resulting dimension.

specific for RDF/OWL. The *Tbox* template has been carefully designed following the structure of the medical protocols defined in the Health-e-Child[5] project for rheumatic patients. Moreover, domain concepts are taken from UMLS. With the previous setup, we are able to generate synthetic instance data of any size and with the intended structural variations to account for heterogeneity and optional values of semantic annotations. In particular, the dataset contains more than half million of instances.

Figure 13 presents how the number of elements in the MD schema (i.e. number of dimensions and measures) affects the time performance to generate the fact table with all valid combinations of instances (i.e. data fact tuples). For the experiment setup we have preselected a set of 11 candidate dimensions and measures from the ontology and have computed all the fact tables that can be generated from all subsets of these 11 elements. The total number of fact tables is $2^{11} = 2048$. Then, we have organized the fact tables according to the number of dimensions and measures in their MD schema (x axis), from two dimensions to eleven. Axis y shows the time performance in seconds. Each boxplot in the figure shows the variance in time between fact tables having the same number of dimensions and measures. The explanation for this variance is that different MD configurations of the same size may obtain very different *CGs* depending on their structural dependencies. Therefore, the number of instances processed

[5] `http://www.health-e-child.org/`

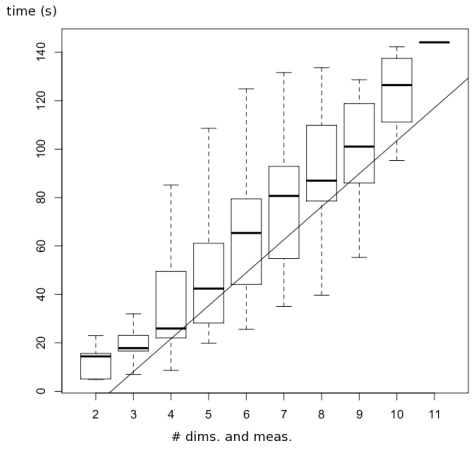

Figure 13: Fact table generation performance w.r.t. the number of dimensions and measures involved.

and the levels of recursion are different, resulting in different processing times. In general, the time complexity increases linearly with respect to the number of dimensions and measures of the fact table, which proves the scalability and efficiency of the approach.

On the other hand, we are also concerned about how the size of the instance store affects the generation of the fact tables. Figure 14 illustrates the results. From the previous experiment we have selected one of the smallest (i.e., 2 elements) and the largest (i.e., 11 elements) MD schema specification. For these two MD configurations we measure the time to create the respective fact tables with instance stores of different sizes, ranging from 100 to 3000 complex instances of type *Patient*. Notice axis x measures the number of subject instances, although the number of total instances in the store ranges from a thousand to more than half million instances. For both configurations, the time performance is linear w.r.t. the size of the instance store, which means the method proposed is scalable.

7.2. Dimensions extraction

In order to measure the quality of the dimension hierarchies obtained with the methods proposed in Section 6 (i.e. global vs. local), we have adapted the measures proposed in [15], namely

$$dilution(D_i, Fragment, T) = \frac{1}{|T|} \sum_{t \in T} \Delta_O(parent_O(t[D_i]), parent_{Fragment}(t[D_i]))$$

$$diversity(D_i, Fragment, T) = \frac{2}{(|T|^2 - |T|)} \sum_{t_1, t_2 \in T, t_1 \neq t_2} \Delta_{Fragment}(t_1[D_i], t_2[D_i])$$

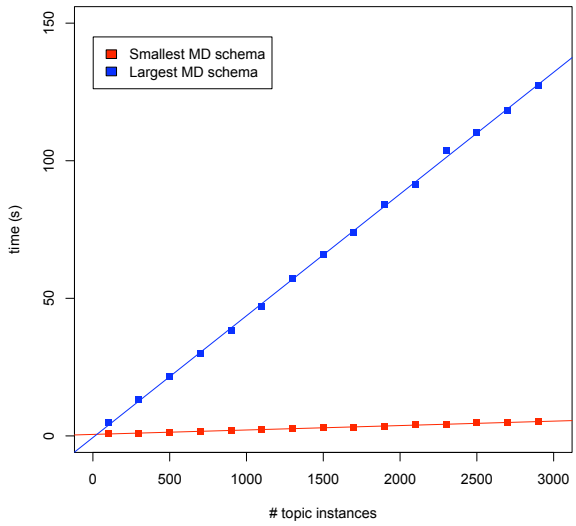

Figure 14: Increase in the time complexity w.r.t. the size of the instance store.

where T is the fact table, $hits(T, n_i)$ is the times n_i has been used in T, $parent_O(n)$[6] is the parent of n in O, and Δ_O is the taxonomic distance between two nodes in O. $t[D_i]$ represents the concept assigned to the fact t for the dimension D_i.

Dilution measures the weighted average distance in the original taxonomy between the new and original parents of the signature concepts from dimension D_i. The weight of each signature concept corresponds to its relative frequency in the fact table. The smaller the dilution, less semantic changes have been produced in the reduced taxonomy. Diversity measures the weighted average distance in the reduced taxonomy of any pair of concepts from dimension D_i used in the fact table. The greater the diversity, better taxonomies are obtained for aggregation purposes. A very low diversity value usually indicates that most concepts are directly placed under the top concept.

We have set up 25 signatures for 14 dimensions of the dataset described in the previous section. The size of these signatures ranges from 4 to 162 concepts (60 on average). The corresponding upper-modules are extracted from UMLS following the method proposed in [29]. The size of these modules ranges from 40 to 911 concepts (404 on average). Their inferred taxonomies present between 8 and 23 levels (17 on average). Results for the global and local methods are

[6]Indeed, we assume that the parent of n in O is the parent in the extracted spanning tree of O.

Measure	Global	Local
Reduction (%)	72.5 - 75.9	76.5 - 79.8
Sig. Lost (%)	7.08-12.12	3.5-7.8
Dilution	0.367-0.473	0.342-0.429
Diversity	9.08-10.79	7.72-9.46

Table 5: Results for global an local dimension extraction methods. Value ranges represent the 0.75 confidence intervals of the results for all the signatures.

shown in Table 5.

From these results we can conclude that: (1) the local method generates smaller dimension tables, (2) the local method implies less signature lost, but dilution values of both methods are not statistically different and, (3) diversity is usually greater in the global method (i.e. richer taxonomies are generated). To sum up, each method optimizes different quality parameters, and therefore their application will depend on the user requirements.

7.3. Implementation

We use *MySQL*[7] database as back-end to store the semantic annotations (i.e. $ABox$), the domain and application ontologies (i.e. $TBox$) and the required indexes. On the other hand, we use the *Business Intelligence* (BI) tool of *Microsoft SQL Server 2008*[8] to instantiate the MD schema designed by the user and create cubes. Our method is completely independent of any data management system. We simply need to create an API to the back-end where the information is stored and the populated MD schema is delivered as a series of tables that can be fed into any off the shelf analysis tool.

In Figure 15 we show the result of one of the MD queries proposed for the use case in section 2.2. In this use case, the user is interested in analyzing the efficacy of different drugs w.r.t. a series of dimensions, such as the disease diagnosed, the patient's age, gender, etc. The method first generates the fact table according to the conceptual MD schema proposed by the analyst and then, for each dimension, a dimension hierarchy is extracted using the *global* approach. The result (i.e. the populated MD schema) has been fed to SQL Server and the BI tool allows the analyst to create cubes and navigate through them. In particular, Figure 15 shows the cube generated by averaging the $damageIndex$ measure by *disease* (rows) and *drug* (columns). As shown, the user can navigate through the different levels of the dimension hierarchies and the measures are automatically aggregated. It is worth mentioning the added value that provides the semantics involved in the aggregations, since the dimension hierarchies express conceptual relations extracted from a domain ontology.

[7] `MySQL: http://www.mysql.com`

[8] `SQL Server 2008: http://www.microsoft.com/sqlserver/2008`

			Level 02 ▾ Level 03			
			⊞ C0003209=anti-inflammatory	⊞ C0021054=Factors,Immunologic	⊟ C2170827=tumornecrosisfactoralphablockers	
					⊞ C0666743=INFLIX	⊞ C1122087=tumornecrosis
Level 02 ▾	Level 03	Level 04	Avg damage Index	Avg damage Index	Avg damage Index	Avg damage Index
⊟ C0003873=Arthritisorpol	⊞ C0409651=rheumatoid/seropositivearthritides		26,10167	26,14971	26,16316	26,50977
	⊞ C0427391=Rheumatoidfactornegative		26,64191	26,93696	27,21934	27,03086
	⊟ C0553662=IdiopathicArth	⊞ C1384600=Juvenilearthritiswithsyste	26,18157	26,48039	26,67122	27,00051
		⊞ C1444840=Juvenileseronegativepoly	26,77114	26,97851	27,29854	27,24215
		⊞ C1444841=Juvenileidiopathicarthriti	27,21066	26,63931	26,25423	26,79112
		⊞ C1444844=Juvenileidiopathicarthriti	27,22942	27,60329	26,49961	26,82511
		Total	26,83747	26,92224	26,67816	26,96601
	Total		26,68180	26,79518	26,69152	26,89885
Total general			26,68180	26,79518	26,69152	26,89885

Figure 15: Example of MD cube created by averaging the $damageIndex$ measure by $disease$ (rows) and $drug$ (columns).

8. Related work

Although there is a significant amount of literature that relates to different aspects of our approach (e.g. SW, MD models, OLAP analysis, etc.), there is little or no research that addresses the analysis of semantic web data (i.e. RDF/(S) and OWL) by using the above-mentioned technologies. However, we find worth reviewing some work on querying and analysis over heterogeneous XML data, which is the standard on which SW languages rely on.

Some research has focused on querying complex XML scenarios where documents have a high structural heterogeneity. In [25, 41] it is shown that XQuery is not the most suitable query language for data extraction from heterogeneous XML data sources, since the user must be aware of the structure of the underlying documents. The lowest common ancestor (LCA) semantics can be applied instead to extract meaningful related data in a more flexible way. [25, 41] apply some restrictions over the LCA semantics. In particular they propose SLCA [41] and MLCA [25] whose general intuition is that the LCA must be minimal. However, in [28, 32] they showed that these approaches still produced undesired combinations between data items in some cases (e.g. when a data item needs to be combined with a data item at a lower level of the document hierarchy). In order to alleviate the previous limitations they propose the SPC (smallest possible context) data strategy, which relies on the notion of closeness of data item occurrences in an XML document. There exists other strategies to extract transactions or facts from heterogeneous XML [38]. However, they are used in data mining applications and they do not care for OLAP properties such as "good" dimensions and summarizability issues. We face similar challenges as the previous approaches mainly due to the structural heterogeneity of the semantic annotations. However, we still need to deal with the semantics, which requires logical reasoning in order to derive implicit information.

Other approaches such as [34, 35] try to incorporate semantics in the design of a data warehouse MD schema by taking as starting point an OWL ontology that describes the data sources in a semi-automatic way. Instead of looking for functional dependencies (which constitute typical fact-dimension relations) in the sources, they are derived from the ontology. However, this work focuses on the design phase, overlooking the process of data extraction and integration to populate the MD schema. This issue is partially addressed in [24] by using SPARQL over RDF-translated data sources. However, this is not appropriate

for expressive and heterogeneous annotations. By combining IE techniques with logical reasoning, in [18] they propose a MD model specially devised to select, group and aggregate the instances of an ontology. In our previous work [31] we define the *semantic data warehouse* as a new semi-structured repository consisting of semantic annotations along with their associated set of ontologies. Moreover, we introduce the *multidimensional integrated ontology* (MIO) as a method for designing, validating and building OLAP-based cubes for analyzing the stored annotations. However, the fact extraction and population is pointed out in a shallow way and it is the main concern of the current paper.

9. Conclusions

More and more semantic data are becoming available on the web thanks to several initiatives that promote a change in the current Web towards the Web of Data, where the semantics of data become explicit through data representation formats and standards such as RDF/(S) and OWL. However, this initiative has not yet been accompanied by efficient intelligent applications that can exploit the implicit semantics and thus, provide more insightful analysis.

In this paper, we investigate how semantic data can be dynamically analyzed by using OLAP-style aggregations, navigation and reporting. We propose a semi-automatic method to build valid MD fact tables from stored semantic data expressed in RDF/(S) and OWL formats. This task is accomplished by letting the user compose the conceptual MD schema by selecting domain concepts and properties from the ontologies describing the data. Dimension hierarchies are also extracted from the domain knowledge contained in the ontologies. The benefits of our method are numerous, however we highlight the following: 1) we provide a novel method to exploit the information contained in the semantic annotations, 2) the analysis is driven by the user requirements and it is expressed always at the conceptual level, 3) the analysis capabilities (i.e. the OLAP-style aggregations, navigation, and reporting) are richer and more meaningful, since they are guided by the semantics of the ontology. To our knowledge, this is the first method addressing this issue from ontological instances. Hence, we do believe this work opens new interesting perspectives as it bridges the gap between the DW and OLAP tools and SW data.

As future work, we plan to improve the method in several aspects. In particular, we plan to extend the local method for generating dimension hierarchies so that the dimension values can be grouped into a defined number of levels or categories. We are also working on an extended conceptual MD specification for the analyst in terms of richer ontology axioms. Regarding performance issues, a promising direction is the application of bitmap indexing techniques to SW data management, including efficient reasoning. Finally, we are also concerned about different SW scenarios where the development of the ontology axioms and the instance store is not coupled. In such scenarios, we need to study how to treat the possible vagueness of the ontology axioms (or even absence) w.r.t. to the instance store, which hinders the proposed extraction of facts.

Acknowledgements

This research has been partially funded by the Spanish Research Program (TIN2008-01825/TIN) and the Health-e-Child integrated EU project. Victoria Nebot was supported by the PhD Fellowship Program of the Spanish Ministry of Science and Innovation (AP2007-02569).

References

[1] GALEN ontology. `http://www.opengalen.org/`.

[2] GO: The Gene Ontology. `http://www.geneontology.org/`.

[3] Linked life data. `http://linkedlifedata.com/`.

[4] Linking Open Drug Data (LODD). `http://esw.w3.org/HCLSIG/LODD`.

[5] MEDLINE: National Library of Medicine's database. `http://www.nlm.nih.gov/databases/databases_medline.html`.

[6] SNOMED CT: Systematized Nomenclature of Medicine-Clinical Terms. `http://www.ihtsdo.org/snomed-ct/`.

[7] The Universal Protein Resource: UniProt. `http://uniprot.org`.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[9] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41:706–716, 2008.

[10] R. Berlanga, E. Jiménez-Ruiz, V. Nebot, D. Manset, A. Branson, T. Hauer, R. McClatchey, D. Rogulin, J. Shamdasani, S. Zillner, and J. Freund. Medical data integration and the semantic annotation of medical protocols. In *CBMS*, pages 644–649. IEEE Computer Society, 2008.

[11] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[12] O. Bodenreider.

[13] O. Bodenreider. Biomedical ontologies in action: role in knowledge management, data integration and decision support. *Yearbook of medical informatics*, pages 67–79, 2008.

[14] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In *ECAI*, pages 298–302, 2004.

[15] K. S. Candan, M. Cataldi, and M. L. Sapino. Reducing metadata complexity for faster table summarization. In I. Manolescu, S. Spaccapietra, J. Teubner, M. Kitsuregawa, A. Léger, F. Naumann, A. Ailamaki, and F. Özcan, editors, *EDBT*, volume 426 of *ACM International Conference Proceeding Series*, pages 240–251. ACM, 2010.

[16] G. Cheng, W. Ge, and Y. Qu. Falcons: searching and browsing entities on the semantic web. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 1101–1102. ACM, 2008.

[17] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User Analysts: An IT Mandate. E. F. Codd and Ass., 1993.

[18] R. Dánger and R. B. Llavori. Generating complex ontology instances from documents. *J. Algorithms*, 64(1):16–30, 2009.

[19] M. dAquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Watson: A gateway for next generation semantic web applications. *Poster, ISWC 2007,*, 2007.

[20] L. Ding, T. Finin, A. Joshi, R. Pan, S. R. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[21] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res. (JAIR)*, 31:273–318, 2008.

[22] C. A. Hurtado, C. Gutierrez, and A. O. Mendelzon. Capturing summarizability with integrity constraints in olap. *ACM Trans. Database Syst.*, 30(3):854–886, 2005.

[23] E. Jiménez-Ruiz, B. Cuenca-Grau, U. Sattler, T. Schneider, and R. Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2008.

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

[25] Y. Li, C. Yu, and H. V. Jagadish. Schema-free XQuery. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 72–83. VLDB Endowment, 2004.

[26] J.-N. Mazón, J. Lechtenbörger, and J. Trujillo. A survey on summarizability issues in multidimensional modeling. *Data Knowl. Eng.*, 68(12):1452–1469, 2009.

[27] M. Mesiti, E. Jiménez-Ruiz, I. Sanz, R. B. Llavori, P. Perlasca, G. Valentini, and D. Manset. Xml-based approaches for the integration of heterogeneous bio-molecular data. *BMC Bioinformatics*, 10(S-12):7, 2009.

[28] T. Näppilä, K. Järvelin, and T. Niemi. A tool for data cube construction from structurally heterogeneous XML documents. *JASIST*, 59(3):435–449, 2008.

[29] V. Nebot and R. Berlanga. Building Tailored Ontologies from very large Knowledge Resources. In *ICEIS Conference Proceedings*, volume 2, pages 144–151. ICEIS, May 2009.

[30] V. Nebot and R. Berlanga. Efficient retrieval of ontology fragments using an interval labeling scheme. *Inf. Sci.*, 179(24):4151–4173, 2009.

[31] V. Nebot, R. Berlanga, J. M. Pérez, M. J. Aramburu, and T. B. Pedersen. Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses. *JoDS XIII*, 5530:1–35, 2009.

[32] T. Niemi, T. Näppilä, and K. Järvelin. A relational data harmonization approach to XML. *J. Inf. Sci.*, 35(5):571–601, 2009.

[33] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Inf. Syst.*, 26(5):383–423, 2001.

[34] O. Romero and A. Abelló. Automating multidimensional design from ontologies. In *DOLAP '07*, pages 1–8, New York, NY, USA, 2007. ACM.

[35] O. Romero, D. Calvanese, A. Abelló, and M. Rodriguez-Muro. Discovering functional dependencies for multidimensional design. In I.-Y. Song and E. Zimányi, editors, *DOLAP*, pages 1–8. ACM, 2009.

[36] I. Sanz, M. Mesiti, G. Guerrini, and R. Berlanga. Fragment-based approximate retrieval in highly heterogeneous XML collections. *Data Knowl. Eng.*, 64(1):266–293, 2008.

[37] K. Skaburskas, F. Estrella, J. Shade, D. Manset, J. Revillard, A. Rios, A. Anjum, A. Branson, P. Bloodsworth, T. Hauer, R. McClatchey, and D. Rogulin. Health-e-child: A grid platform for european paediatrics. *Journal of Physics: Conference Series*, 119, 2008.

[38] A. Tagarelli and S. Greco. Semantic clustering of XML documents. *ACM Trans. Inf. Syst.*, 28(1), 2010.

[39] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364, 2010.

[40] H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, and Y. Pan. Semplore: A scalable ir approach to search the web of data. *Web Semant.*, 7:177–188, September 2009.

[41] Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 527–538, New York, NY, USA, 2005. ACM.

Victoria Nebot received her B.S. degree in Computer Science from Universitat Jaume I, Spain, in 2007. She joined the Temporal Knowledge Bases Group (TKBG) at Universitat Jaume I as a PhD Student in 2008. Her main research is focused on analyzing and exploiting semi-structured and complex data derived mainly from the Semantic Web. In particular, she is interested in applying data warehousing and OLAP techniques that enable to analyze semantic data for decision support system tasks and the application of these techniques to the biomedical domain.

Rafael Berlanga is associate professor of Computer Science at Universitat Jaume I, Spain, and the leader of the TKBG research group. He received the B.S. degree from Universidad de Valencia in Physics, and the PhD degree in Computer Science in 1996 from the same university. In the past, his research was focused on temporal reasoning and planning in AI. His current research interests include knowledge bases, information retrieval and the semantic web. He has directed several research projects and has published in several journals and international conferences in the above areas.