MULTIDIMENSIONAL INTEGRATED ONTOLOGIES: A FRAMEWORK FOR DESIGNING SEMANTIC DATA WAREHOUSES

Victoria Nebot, Rafael Berlanga, Juan Manuel Pérez, María José Aramburu
Universitat Jaume I
Av. Vicent Sos Baynat, s/n
E-12071 Castelló, Spain
{romerom, berlanga, juanma.perez, aramburu}@uji.es

Torben Bach Pedersen
Aalborg University
Selma Lagerløfs Vej 300,
DK-9220 Aalborg Ø, Denmark
tbp@cs.aau.dk

Abstract. The Semantic Web enables companies and organizations to gather huge amounts of valuable semantically annotated data concerning their subjects of interest. Nowadays, many applications attach metadata and semantic annotations taken from domain and application ontologies to the information they generate. From our point of view, the concepts in these ontologies could describe the facts, dimensions, categories and values implied in the analysis subjects of a data warehouse. In this paper we propose the Semantic Data Warehouse to be a repository of ontologies and semantically annotated data resources. We also propose an ontology-driven framework to design multidimensional analysis models for Semantic Data Warehouses. This framework provides means for building an integrated ontology, called the Multidimensional Integrated Ontology (MIO), including the classes, relationships and instances that represent interesting analysis dimensions and measures. The reasoning capabilities of a MIO can be used to check the properties required by current multidimensional databases (e.g., dimension orthogonality, category satisfiability, etc.). In this paper we also sketch how the instance data of a MIO can be translated into OLAP cubes for analysis purposes. Finally, some implementation issues of the overall framework are discussed.

Keywords: Data warehouses, Semantic Web, Multi-ontology integration

1. Introduction

The Semantic Web is a rich source of knowledge whose exploitation will open new opportunities to the academic and business communities. One of these opportunities is the analysis of information resources for decision support tasks such as the identification of trends, and the discovery of new decision variables. Semantic annotations are formal descriptions of information resources which usually rely on widely accepted domain ontologies. The main reason for using domain ontologies is to set up a common terminology and logic for the concepts involved in a particular domain. Semantic annotations are especially useful for describing unstructured, semi-structured and text data, which cannot be managed properly by current database systems. Nowadays many applications (e.g., medical applications) attach metadata and semantic annotations to the information they produce, for example medical image, laboratory tests, etc. In the near future, large repositories of semantically annotated data will be available, opening new opportunities for enhancing current decision support systems.

Data warehouse systems are stores of information aimed at analysis tasks. This information is extracted from existing databases and is pre-processed to harmonize its syntax and semantics. Thus, one of the main purposes of data warehouse systems is the integration of information coming from several sources. Afterwards, OLAP systems can be applied to efficiently exploit the stored information. Both types of systems rely on multidimensional data models, which distinguish the stored measures from the analysis dimensions that characterize them.

In this paper we tackle the problem of combining data warehouse and Semantic Web technologies. Our proposal is a framework for designing multidimensional analysis models over the semantic annotations stored in a Semantic Data Warehouse (SDW). In our approach, an SDW is conceived as a XML repository that includes web resources, domain ontologies and the semantic annotations made with them. Being a data warehouse, this repository is subject oriented, and therefore it is aimed at recording only data that is relevant for specific analysis tasks.

Our work is being carried out in the context of a larger research project about the integration and exploitation of biomedical data provided by clinicians for research tasks. The framework presented here is based on the specification of a Multidimensional Integrated Ontology (MIO) over the SDW ontologies in order to retrieve the ontology classes and instances that will later be used in the multidimensional analysis. To our best knowledge, our approach is the first one on addressing the following requirements:

- *Multi-ontology design*. Much semantic data is generated in the context of very complex scenarios involving several domain ontologies. The framework proposed in the paper allows the selection of the concepts needed for the analysis through different ontologies.

- *Scalability*. As domain ontologies usually have a considerably large size, the method for building MIOs must be scalable. We will achieve these scalability requirements by extracting only those modules or fragments that are necessary from the source ontologies.

- *Formally well-founded approach.* In order to keep the semantics and inference mechanisms of the source ontologies, the proposed design process relies on formalisms that have been widely accepted for the Semantic Web (e.g., Description Logics).

The main contributions of the paper can be summarized as follows:

1. A framework for designing and building Semantic Data Warehouses.

2. An application scenario and a running use case to establish the requirements and to illustrate the usefulness of our techniques.

3. A methodology for the design, automatic generation and validation of Multidimensional Integrated Ontologies. By integrating the concepts and properties of several ontologies coming from the same application domain, a MIO establishes the topics, measures, dimensions and hierarchies required by a specific data analysis application.

4. The automatic construction of a multidimensional cube, according to the specifications of a MIO, starting from the annotated data stored in the SDW, in order to allow the analysis of this data by using traditional OLAP operators.

5. The study of several alternatives for implementing the proposed SDW.

The rest of the paper is organized as follows. Section 2 describes an application scenario that motivates our approach. Section 3 reviews the related work including: Description Logics, OWL and OLAP; the existing approaches to annotate biomedical data; the combination of Semantic Web and data warehouse technologies; and different alternatives for exploiting knowledge from multiple ontologies. Section 4 introduces our approach to a Semantic Data Warehouse. Section 5 explains the methodology proposed for designing Multidimensional Integrated Ontologies and Section 6 gives some implementation guidelines. Finally, Section 7 presents some conclusions and future work.

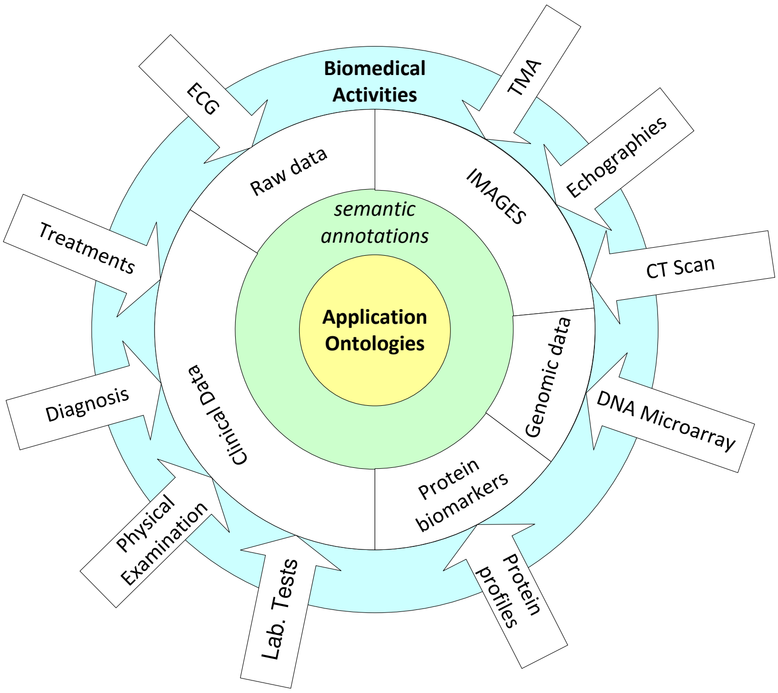

Figure 1: Generation of semantic annotations in the biomedical domain.

2. Application Scenario & Use Case

In this section we describe an application scenario for an SDW along with a use case that will serve to define the examples of the rest of the paper. By defining this application scenario, we will identify a list of requirements that can be considered common to many applications of SDWs, and that, therefore, can be applied to prove the usefulness of the framework proposed in this paper.

Our application scenario is Biomedicine in which, at the moment, vast amounts of semantically annotated data are being generated by many different types of data management systems (see section 3.2). In order to guide the process of semantically annotating the data, current data management systems adopt specific application ontologies relying on one or more widely accepted domain ontologies. A domain ontology is a very large corpus of semantically related data that describe the knowledge and vocabularies agreed by the relevant biomedical community. The reader can find a good review of the main biomedical ontologies in (Rubin et al., 2007).

Figure 1 shows the usual process of generating semantic annotations for the data elements that biomedical activities produce. The application ontologies that rule the structure of the semantic annotations are located in the core of the data management system. At the cortex part, we find the different types of complex data elements, coming from very different biomedical activities and departments, that need to be annotated before being exploited in the context of an SDW. Typically, semantic annotations are expressed in XML or RDF formats.

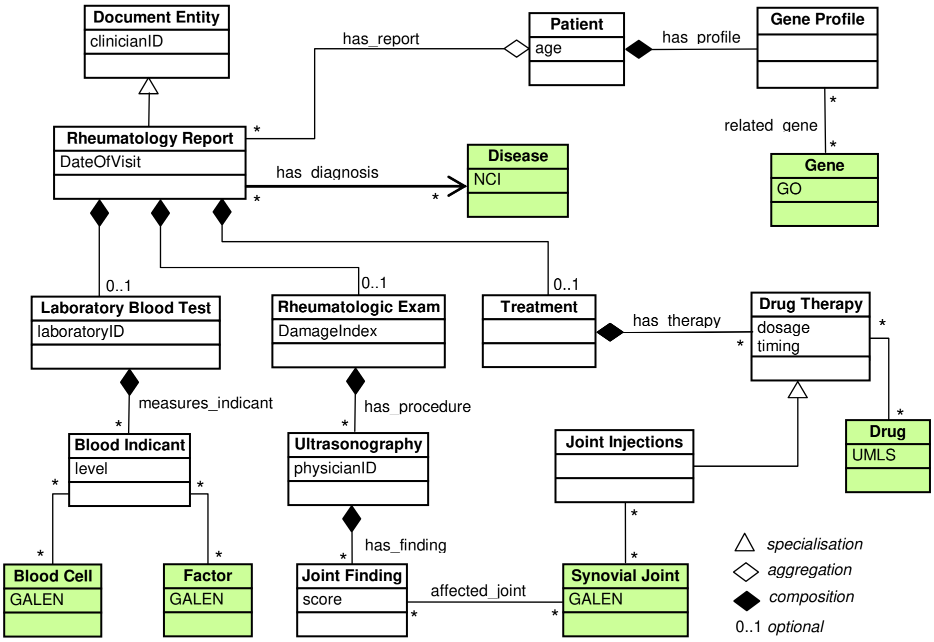

Figure 2. A fragment of an application ontology for Rheumatology.

In the biomedical scenario, semantically annotated data consists of many different types of data (e.g. lab test reports, ultrasound scans, images, etc.) originating from heterogeneous data sources. This data also presents complex relationships that evolve rapidly as new biomedical research methods are applied. As a consequence, this data cannot be properly managed by current data warehouse technology, mainly because it is complex, semi-structured, dynamic and highly heterogeneous.

Figure 2 illustrates an ontology fragment for the Rheumatology domain. As the figure shows, a patient may have different rheumatology reports, authored by some clinicians, consisting of the results of some blood tests and rheumatologic exams, the diagnosis of a disease (defined in the domain NCI ontology) and the proposed treatment. The objective of these examinations is to estimate an overall damage index by performing some ultrasonography tests. The treatment is modelled as a collection of drug therapies, sometimes applied in the affected joints. The joint set is compiled from the GALEN domain ontology. The patient has a genetic profile. The cells and genes involved in the genetic profiles are described by the GALEN and GO domain ontologies, respectively.

Although in Figure 2 we have used UML to graphically represent the ontology fragment, the actual representation formalism will in practice rely on standard languages such as RDF/S and OWL. External concepts coming from domain ontologies are represented in the UML diagram with shaded boxes, indicating the source ontology within the attribute section (e.g. NCI, GO, etc.). Domain ontologies can be used to control the vocabulary and to bring further semantics to the annotated data. Table 1 shows an example of semantically annotated data generated from the application ontology of Figure 2 and stored as RDF triples.

Subject	**Predicate**	**Object**
Patient8991u	type	Patient
Patient8991u	age	15
Patient8991u	sex	Male
Patient8991u	has_report	RR001u
RR001u	type	Rheumatology
RR001u	dateOfVisit	2008/02/22
RR001u	clinicianID	Clinician2293u
RR001u	has_Diagnosis	RA1
RA1	label	Rheumatoid_Arthritis^{NCI}
RA1	type	Disease^{NCI}
RR001u	has_Section	LBT1234u
…	…	…

Table 1: Application ontology instances stored as RDF triples.

In the context of this application scenario, our aim is to build a warehouse where semantically annotated data can be analysed with OLAP-based techniques. As use case, we propose to analyse the efficacy of different drugs in the treatment of several types of inflammatory diseases, mainly rheumatic ones. The analysts of this use case should define the dimensions, measures and facts that will allow the analysis of the semantic annotations, gathered from several hospitals and, therefore, expressed with different application ontologies. Notice that at this point, the analyst does neither know the values nor the roll-up relationships that will eventually be used in the resulting cube. As we will show, the framework presented in this paper will capture this information from the application and the domain ontologies involved in the analysis.

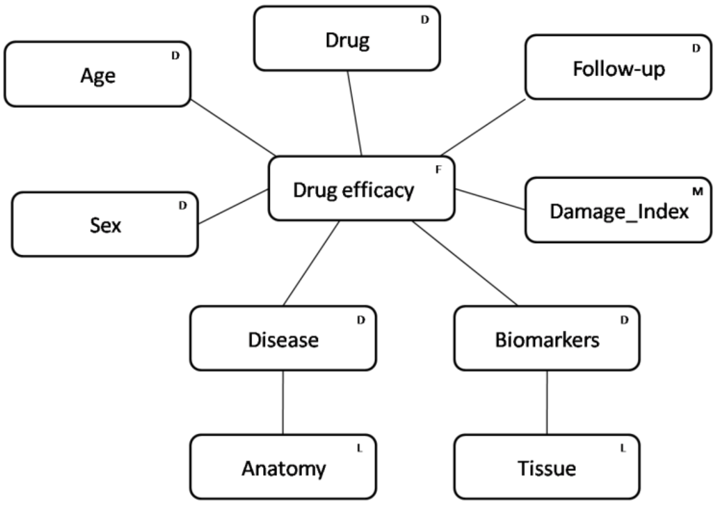

Figure 3: Dimensions defined for analyzing rheumatology patients. We use the letter *D* for dimensions, *F* for facts, *M* for measures and *L* for dimension levels.

Figure 3 shows the seven dimensions that we have selected in order to study this use case from different points of view, including: the patient's age and gender, the subtype of disease (diagnosis), the biomarkers taken from the patient, the damage index of patient´s joints and the drugs administered during the follow-up visits of the patient. Since we consider that the relation that exists between disease symptoms and affected body parts is very relevant for the analysis, we have introduced the category Anatomy in the disease dimension. The biomarkers of interest include blood cells, blood factors and genes. The category Tissue has been similarly introduced in the biomarkers dimension in order to relate biomarkers with their associated tissues.

In this use case, OLAP technologies can be applied to perform useful analysis operations over the gathered data, as for example:

- By applying roll-up operations, we can aggregate data into coarser granularities such as drug families, active principles, types of diseases, and so on. On the contrary, by means of the available drill-down operations, we can refine each of the analysis dimensions to obtain data with a finer granularity. This kind of operations can give useful information to the clinicians about the relation between diagnosis and treatment efficacy.
- By applying selection and projection operations, we can restrict the analysis to patient subsets according to criteria based on age, sex, affected body parts, etc.

In this section we have defined an application scenario and a use case for the SDWs we want to achieve. In this scenario we identify the following set of application requirements:

1. Integration of biomedical data, information and knowledge to gain a comprehensive view of patients.
2. Scalable data storage functionalities to store the collected semantic information as well as the relevant application and domain ontologies.
3. Flexible ways of specifying analysis dimensions, measures and facts based on medical criteria.
4. Easy exploration of large domain ontologies considering their implicit semantics, and the possible overlapping in their concepts (e.g. mappings).

In the context of other application scenarios these requirements should not be much different, so from our point of view, they can be considered as a basic set of requirements for a generic analysis application of an SDW. It is worth mentioning that the contributions of this paper described in the introduction are aimed at covering all these requirements.

3. Background and Related Work

In this section we review the basic concepts involved in the representation, generation and storage of semantic annotations of data, as well as some related work about the analysis of semantic data.

3.1. OWL, Description Logics and OLAP

The Ontology Web Language (OWL) is a language for the specification of ontologies, whose definition by the W3C Consortium has empowered the biomedical community to develop large and complex ontologies like the NCI thesaurus, GALEN, etc. OWL provides a powerful knowledge representation language that has a clean and well defined semantics based on Description Logics (DL). Description Logics are a family of knowledge representation formalisms devised to capture most of the requirements of conceptual modelling. These formalisms are decidable subsets of First Order Logic that are expressive enough to capture interesting conceptual modelling properties. The main purpose of DLs is to provide a formal theory that can be used to validate conceptual schemata (Franconi & Ng, 2000) of heterogeneous databases (Mena et al., 2000), data warehousing design and multidimensional aggregation modelling (Baader & Sattler, 2003). It is worth mentioning that Baader & Sattler (2003) and Franconi & Ng (2000) apply DLs in the context of a traditional warehouse. Our proposal is different; we propose to design the warehouse starting from a collection of semantically annotated data. We use DLs for helping the warehouse designer to transform ontology fragments into analysis dimensions, by testing if these dimensions satisfy a set of properties desirable for OLAP applications.

Let us briefly introduce the basic constructors of Description Logics through the basic language $\mathcal{ALC}$ (Schmidt-Schauss & Smolka, 1991), which is summarised as follows:

$$\mathcal{ALC} ::= \bot \mid A \mid C \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$$

The basic elements of $\mathcal{ALC}$ are concepts (classes in OWL notation), which can be either atomic (A) or derived from other concepts (expressions C and D). Complex concepts are built by using the classical Boolean operators over concepts, namely: *and* ($\sqcap$), *or* ($\sqcup$) and *not* ($\neg$). Value restrictions on the concept individuals (instances in OWL notation) are represented through roles (object properties in OWL notation), which can be either existential ($\exists R.C$) or universal ($\forall R.C$). The universal concept is denoted with $\top$, whereas the empty concept is denoted with $\bot$. The empty concept is usually associated with inconsistencies and contradictions in the ontology.

Currently there exist several reasoners that deal with some Description Logic languages[1], although most of them do not fully support the retrieval of large sets of asserted instances. Indeed, the complexity of these reasoners is PSpace-complete, which does not guarantee scalability for large domains.

Additionally, several DL constructors have been proposed to capture the main elements of conceptual modelling for databases. For example, *concrete domains* were introduced to account for the usual data types in a conceptual database schema. It has been demonstrated that domains like the integers and strings can be easily introduced into a DL without losing decidability[2] (Lutz et al., 2005). Furthermore, users can state features (i.e., relations between instances and values from these domains) with predicates expressing value comparisons. OWL languages support these constructors via the so-called data type properties. Another interesting constructor for OLAP applications is that of role composition, $R \circ P$, which recently has been introduced in OWL. Role composition allows us to express joined relationships making the intermediate involved concepts implicit. Reasoning over role compositions has been shown to be decidable (Horroks & Sattler 2003), but it is not fully supported by current reasoners yet.

Concerning data warehouse operations, Baader & Sattler (2003) introduced aggregates over concrete domains. The resulting language is called $\mathcal{ALC}(\Sigma)$, and extends the basic language $\mathcal{ALC}$ with concrete domains and a limited set of aggregation functions, namely: sum, min, max and count. Aggregates are introduced through complex features of the form $\Gamma(R \circ u)$, which relate each instance with the aggregate Γ over all the values reachable from R followed by the feature u. For example, we can define the following complex feature sum(month ∘ income) to relate instances with their annual

[1] See http://www.cs.man.ac.uk/~sattler/reasoners.html for an exhaustive list. More information about DLs can be found at http://dl.kr.org/.

[2] This occurs whenever the introduced domain satisfies the so-called *admissibility* property.

incomes. With this complex feature we can ask for employees having annual incomes greater than 100,000 Euros by means of the concept:

$$\text{Employee} \sqcap \exists \text{year}.{>}(\text{sum}(\text{month} \circ \text{income}), 100000)$$

However, DLs formalisms present important limitations for representing complex measures and aggregations. Baader & Sattler (2003) also demonstrate that handling aggregates in DLs usually leads to undecidability problems, even for very simple aggregates such as sum and count. Moreover, decidable cases present a level of computational complexity too high for practical real-world applications. Baader and Sattler indicate that some interesting inference problems for multidimensional models, such as summarizability, have not been treated by the proposed DLs. Finally, there are no reasoners able to deal with the advanced features required by these new constructors.

Because of these reasons, we propose a new framework to define an integrated ontology that will be used to build a multidimensional data schema over which to apply the OLAP operations required by the analysis tasks. In this way, summarizability will be ensured by building a valid cube from this multidimensional schema so that aggregations are performed over it, out of the DL formalism.

3.2. Annotating biomedical data

In the biomedical scenario there exist a large number of initiatives for annotating biomedical databases for the Semantic Web. For example, in the SEMEDA project, Köhler et al. (2003) use a controlled vocabulary and an RDF-like ontology to annotate tables, attributes and their domains to derive cross-references between databases. ONTOFUSION (Pérez-Rey et al., 2005) is another approach based on the integration of local conceptual schemata into a global biomedical ontology. A good review of semantic-based approaches for biomedical data integration can be found in (Louie et al., 2006). It is worth mentioning that most of the current work in biomedical applications uses OWL as the representation language for ontologies and semantic annotations.

Currently, there are several ongoing international projects that are aimed at the interchange of massive biomedical data, for example caBig[3], openEHR[4] and Health-e-Child[5] to mention a few. These projects also concern the semantic annotation of data through well-established biomedical ontologies.

Other previous works propose to use OLAP techniques to analyse biomedical data. In (Wang et al. 2005) OLAP operations are applied to discover new relations between diseases and gene expressions as well as to find out new classification schemes for patients. They also propose the use of well-known domain ontologies (e.g., GO[6] for classifying genes and OpenGalen[7] for classifying diseases) to define analysis dimensions. However, the authors do not explain how these ontologies can be translated into OLAP dimensions and how factual data can be semantically annotated for analysis.

From all the previous works and projects, three logical data layers can be identified for the application scenario, namely: the domain ontologies, the data schemata and the generated data. All this data and knowledge pieces are eventually expressed in XML, using the different standards best suited for each layer: RDF/S and OWL for the first one, RDF/S and XML Schema for the second one, and XML for the

[3] caBig project: https://cabig.nci.nih.gov/

[4] openEHR project: http://www.openehr.org/

[5] Health-e-Child project: http://www.health-e-child.org/

[6] GO (Gene Ontology): http://www.geneontology.org/

[7] Galen ontology: http://www.opengalen.org/open/crm/crm-anatomy.html

third one. We also follow this logical structure in our approach to designing an SDW (see Section 4).

3.3. Data Warehousing and Semantic Web Technologies

In this section we review the work that combines data warehouse and Semantic Web technologies. We start with two papers that extend the functionality of a data warehouse with Semantic Web technologies, and then we consider previous works on analysing semantic data with multidimensional data models.

Priebe & Pernul (2003) propose to use a global ontology to annotate OLAP reports and other Web resources such as textual documents. Then, users can contextualise OLAP reports by retrieving the documents related to the metadata (search keywords) attached to them. Here, the global ontology is expressed in RDF/S and it contains domain-specific information along with the values of the hierarchies used in the OLAP database.

Skoutas & Simitsis (2006) work on the automation of the data warehouse's ETL process by applying Semantic Web technologies. They propose to build an ontology that uses OWL constructs to describe and relate the source and target data source schemata. Afterwards, a reasoner is used for identifying the sequence of operations needed to load the warehouse. In a more recent paper, Simitsis et al. (2008) present a template-based natural language generation mechanism to transform both the formal description of the data sources expressed in the ontology, and the inferred ETL operations into a narrative textual report more suitable for the user.

The works by Priebe & Pernul (2003) and Skoutas & Simitsis (2006) apply the Semantic Web infrastructure to extend the functionality of the "traditional" data warehouses, but they do not address the analysis of data gathered from semantic sources. In contrast, our proposal consists of a method for designing multidimensional analysis models over the semantic annotations stored in the SDW. To the best of our knowledge, there are only two recent papers aimed at analysing semantic data with multidimensional models, (Romero & Abello, 2007) and (Danger & Berlanga, 2008).

Romero & Abelló (2007) address the design of the data warehouse multidimensional analysis schema starting from an OWL ontology that describes the data sources. They identify the dimensions that characterize a central concept under analysis (the fact concept) by looking for concepts connected to it through one-to-many relationships. The same idea is used for discovering the different levels of the dimension hierarchies, starting from the concept that represents the base level. In this work the input ontology indicates the multiplicity of each role in the relationships; and a matrix keeps, for each concept, all the concepts that are related by means of a series of one-to-many relationships. The output of the Romero & Abelló's method is a star or snowflake schema that guaranties the summarizability of the data, suitable to be instantiated in a traditional multidimensional database. The application of this work is valid in scenarios where a single ontology of reduced size, with multiplicity restrictions, is used for annotating the source data. However, as discussed in Section 2, a real application will usually involve different domain ontologies of considerable large size; and unfortunately, the multiplicity information is rarely found in the source ontologies.

Danger & Berlanga (2008) propose a multidimensional model specially devised to select, group and aggregate the instances of an ontology. The result of these operations is a set of tuples, whose members are instances of the ontology concepts. They also present the adaptation of a feature selection algorithm to discover interesting potential analysis dimensions. This algorithm builds the dimension hierarchies by selecting the relationships in the ontology that maximize the information gain. Like

Romero & Abelló (2007), Danger & Berlanga only consider scenarios with a single ontology.

As it can be observed, both papers are more concerned with the extraction of interesting dimensions from isolated ontologies rather than analysing a large set of stored SDW annotations. Moreover, in a real-world scenario, the SDW can contain annotations defined in several large inter-linked ontologies. Our contribution in this context is twofold. First, we define the Semantic Data Warehouse as a new semi-structured repository consisting of the semantic annotations along with their associated set of ontologies. Secondly, we introduce the Multidimensional Integrated Ontology as a method for designing, validating and building OLAP-based cubes for analysing the stored annotations.

The development of the Semantic Web relies on current XML technology (e.g, XML Schemas and Web Services). In Perez et al. (2008), we surveyed the combination of XML and data warehouses. The work on the construction of XML repositories (Xyleme; 2001) is particularly relevant to the SDW, since the ontologies and their instance data are typically expressed in XML-like formats. Xyleme (2001) addresses the problems of gathering, integrating, storing and querying XML documents. In order to deal with the high level of dynamicity of web data sources, the Xyleme system allows users to subscribe to changes in an XML document (Nguyen et al., 2001), and applies a versioning mechanism (Marian et al., 2001) to compute the differences between two consecutive versions of an XML document.

However, XML techniques for change control are not useful for ontologies, as we must keep track of non-explicit (i.e. inferred) semantic discrepancies between versions. Although some preliminary tools exist, like OWLDiff[8]), further research must be carried out to study the impact of these changes in the SDW design and its derived OLAP cubes. In this paper we will not treat ontology versioning as it is out of its scope. Thus, we assume that the ontologies stored in the SDW are static.

3.4. Multi-ontology scenarios

The application scenario presented in this paper reveals new data acquisition tools being applied in the biomedical domain. These tools are increasingly incorporating ontology services that allow end-users (e.g. clinicians) to properly annotate data in a standard and controlled way. This task is fulfilled by browsing and selecting terms from domain ontologies and vocabularies (Garwood et al., 2004, Jameson et al., 2008). In order to integrate and analyze the large amounts of semantic annotations generated by these tools, we propose the construction of a MIO that gathers only the *right amount* of knowledge from the different domain ontologies that were used to annotate the data.

A lot of research works have dealt with multi-ontology scenarios, which is the key feature of a distributed environment like the Semantic Web. For example, terms and works encountered in the literature which claim to be relevant include: mapping, alignment, merging, articulation, fusion, integration and so on (Kalfoglou and Schorlemmer, 2003). The scope of this paper is not to provide a new framework for ontology integration and mapping. Instead, we propose the construction of MIOs specifically designed to meet the requirements and restrictions of the application scenario presented. However, since there is an extensive literature concerning ontology modularization and mapping, we will highlight the main approaches devised to deal with several ontologies along with their suitability for our application scenario. Finally, we will justify the approach followed to build our MIO framework.

[8] OWLDiff: http://sourceforge.net/projects/owldiff

OBSERVER (Mena et al., 2000) and OIS (Calvanese et al., 2001) are some of the first approaches that tackle the problem of semantic information integration between domain specific ontologies. The former system is based on a query strategy where the user specifies queries in one ontology's terms and then these queries are expanded to other ontologies through relationships such as synonymy, hyponymy and hypernymy. The latter also uses the notion of queries which allow for mapping a concept in one ontology into an integrated view. However, these approximations are not suitable for our application scenario since our aim is to construct a new stand-alone ontology composed by pieces or fragments from several ontologies. Therefore, we have studied the developments in modular ontologies, since they seem to suit better our purposes.

E-connections (Grau et al., 2005) is a formalism that was designed for combining different logics in a controlled way. It introduces a new family of properties called "link" properties which are associated with domains (component ontologies). Each domain can declare which foreign ontologies it links to. However, E-connections do not allow the specification of subsumption relationships between concepts coming from different ontologies and it works only under disjoint domains. Moreover, E-connections is carried out by extending OWL with new non-standard syntax and semantics. The Distributed Description Logics (DDL) formalism (Borgida and Serafini, 2003) provides mechanisms for referring to ontology concepts and for defining "bridge rules" that encode subsumption between concepts of different ontologies. Context OWL (C-OWL) (Bouquet et al., 2003) is an extension of DDLs that suggests several improvements, such as a richer family of bridge rules, allowing bridging between roles, etc. C-OWL also extends OWL syntax and semantics. In contrast, in C-OWL it is not allowed to reuse foreign concepts in restrictions as in E-connections. There is yet another approach called Package-based Description Logics (P-DL) (Bao et al., 2006) that tries to overcome the limitations introduced by E-connections and C-OWL by allowing both subsumption between different ontologies, and foreign concepts in restrictions. However, as in the above mentioned approaches, another non-standard syntax and semantics is introduced and reasoning support is very restricted.

In all previous approaches we can observe serious limitations that prevent us from using them in the construction of our MIO framework. In first place, they all introduce changes to the syntax and semantics of OWL, therefore, all the available infrastructure such as OWL parsers and reasoners would need to be extended. Moreover, they severely restrict reuse by other organizations and only accept customized, non-standard toolsets. Concerning reasoning aspects, reasoning with multiple distributed ontologies can arise some problems with respect to completeness and performance. Completeness depends on the availability of each local reasoner, which in a distributed network could be unreachable. Moreover, the communication costs between nodes in the system can become a bottleneck, since communication problems can arise. Borgida and Serafini (2003) also establish a connection between DL and DDL that allow them to transfer theoretical results and reasoning techniques from the classical DL literature under certain circumstances. Unfortunately, their approach to construct a global DL ontology implies copying all the axioms of the local ontologies. In our application scenario, this approach is not scalable since domain ontologies are usually very large and complex. In order to address the problems of previous approaches, Stuckenschmidt and Klein (2007) define modular ontologies in terms of a subset of DDL and provide rationales for the restrictions applied. They compute subsumption relations between external concepts offline and store them as explicit axioms in the local ontologies. However, this modular approach can be computationally very expensive because in the worst case it has exponential cost.

We address the previous limitations by proposing the use of alternative techniques to extract fragments and modules from ontologies and combine them in the resulting MIO framework, namely: OntoPath (Jiménez-Ruiz et al., 2007) and Upper Modules (UM) (Jiménez-Ruiz et al., 2008). The application of these tools provides a viable alternative without changing the current Semantic Web infrastructure. In this way, ontologies can be expressed using standard OWL syntax and semantics, and external tools implementing different modularization algorithms extract a fragment or module according to the specific requirements of the target application. As a result, module extraction algorithms do not require any change to the OWL semantics. Moreover, we overcome the scalability problems that may arise when reasoning with several large ontologies by building a MIO that only comprises the relevant knowledge (e.g. relevant modules or fragments). Both techniques will be further explained in Section 5.3.

4. An Approach to Semantic Data Warehouses

We conceive a Semantic Data Warehouse as a semi-structured data warehouse that stores ontology-based semantic annotations along with the mechanisms that allow the execution of analysis operations over the stored data. The special features of this kind of semantically-rich data will require the application of OWL and general XML technologies when building and managing the warehouse.

In Figure 4, we can distinguish several components of the framework proposed for designing and analysing the SDW. As we have already stated, the core part of the framework uses the SDW ontologies to specify a Multidimensional Integrated Ontology suitable for analysis purposes. In the left side of the figure, we can see the processes in which the user of the framework (e.g. analyst) actively participates during the design of the MIO. In the centre of the figure, we show the tools needed to come up with the MIO and with the subsequent multidimensional cube. Finally, the right side of the figure shows the logical organization of the data and the schemata of the SDW. We will begin by explaining the latter.

In a real-world scenario, an SDW requires storing the huge amount of annotated data to be analysed together with the application ontologies used to generate it. However, given the complexity of many applications, application ontologies are usually based on one or more community-agreed ontologies, also denoted domain ontologies, which should also be part of the warehouse. In this way, the resulting SDW would include all the data and knowledge necessary for processing complex analysis queries.

The four types of data sets that an SDW stores and their relationships (right side of Figure 4) are explained in turn:

- A set of **domain ontologies** that will contain the agreed terminology and knowledge about the subject of analysis. In our biomedical scenario, this set consists of the ontologies that could be useful for annotating patient data, such as UMLS, NCI Ontology, etc.

- A set of **application ontologies** needed for generating the data that will be stored in the intended data warehouse. These ontologies resemble database schemata but they are more flexible in the sense that they allow incomplete, imprecise and implicit definitions for the generated data. These ontologies will use the domain ontologies to bring proper meaning to their concepts. In a real-world scenario, application ontologies should be tailored to the requirements of the users that will share activities over the generated data. For example, in the biomedical scenario, the application ontology defined by Rheumatology clinicians will be quite different from that defined by Cardiology specialists.

- The set of **ontological instances** generated from the previous application ontologies. This constitutes the main repository of the data warehouse, and it is assumed to be the largest part of it. The analysis of ontological instances is the main purpose of the SDW, and new tools able to process complex analysis operations over them need to be developed.

- The set of **MIO ontologies** generated during the design process. These ontologies are the core feature of the SDW. They gather together only the relevant external knowledge so that later analysis can be performed over the ontological instances. MIOs can be thought as alternative analysis perspectives over the ontological instances. Further details about their definition, generation and validation are given in the next section.

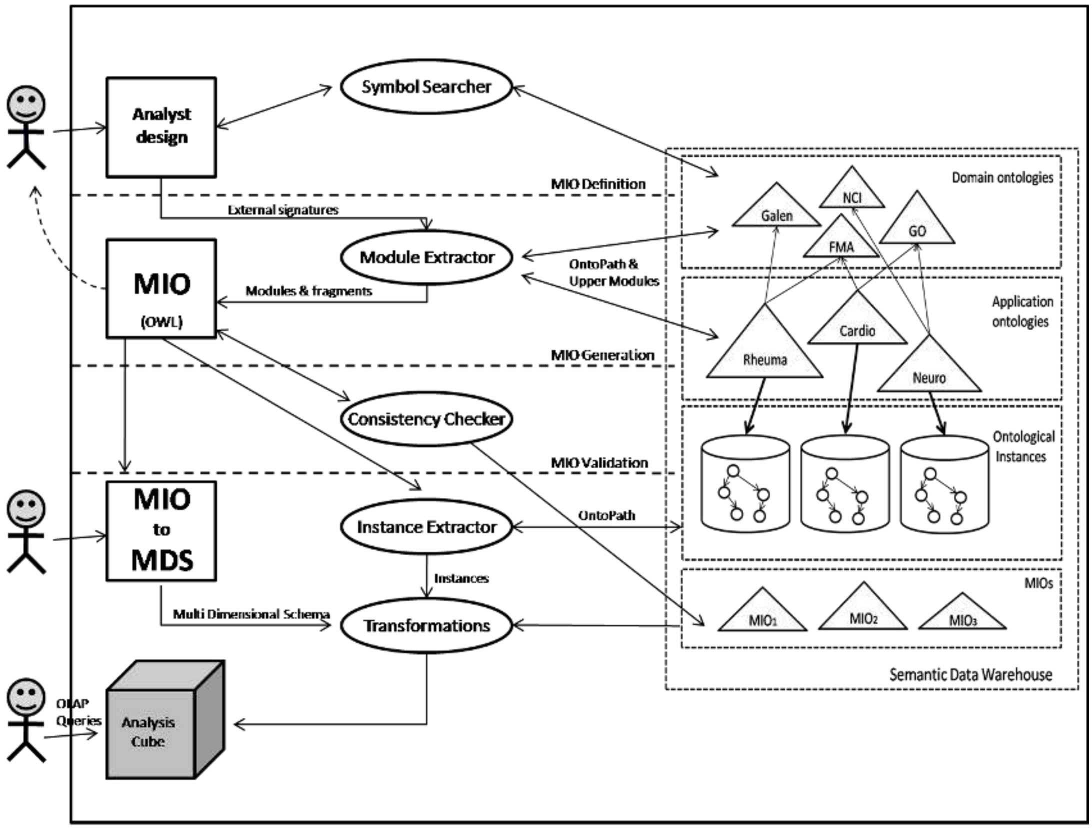

Figure 4. Proposed framework for the design of SDWs.

In order to generate the MIOs for the SDW, we also need a set of mappings between the ontologies whose domains overlap. This is necessary because different application ontologies can be using different domain ontologies to denote similar concepts, for example NCI or Galen for disease concepts. It is also possible that the analyst specifies dimensions with category levels that involve different domain ontologies. Therefore, we need mechanisms that reconcile the overlapping concepts borrowed from different ontologies.

In our work, we represent mappings as 7-tuples $\langle id, s_1, s_2, O_1, O_2, \mathrm{R}, \phi\rangle$, where id is the unique identifier of the mapping, s_1 and s_2 are symbols from ontologies O_1 and O_2 respectively, R is the mapping relationship between these symbols, namely: equivalence

(≡), subsumption (⊑) and disjointness (⊥), and ϕ is a confidence value in the range [0,1], which is usually estimated by the tool that discovered the mapping. From now on, an ontology symbol *s* that is transformed from the ontology *O1* to *O2* by using a mapping *m*, is denoted with $s^{O1}{}_{m}{}^{\rightarrow O2}$.

The next section is completely devoted to describing the framework presented in Figure 4, which comprises the workflow for building the MIO and performing analysis operations over the stored data.

5. Multidimensional Integrated Ontologies (MIOs)

In this section we describe a technique for defining multidimensional conceptual schemata as a first step to analyze SDWs. A Multidimensional Integrated Ontology (MIO) can be considered as a customized ontology whose concepts and roles represent dimensions, categories, measures and facts. This ontology must also include all the axioms and assertions necessary for validating the intended multidimensional data model. As a result, MIOs can be used for both guiding designers in the definition of the analysis dimensions, and checking the resulting model for some interesting properties which ensure that valid final cubes will result.

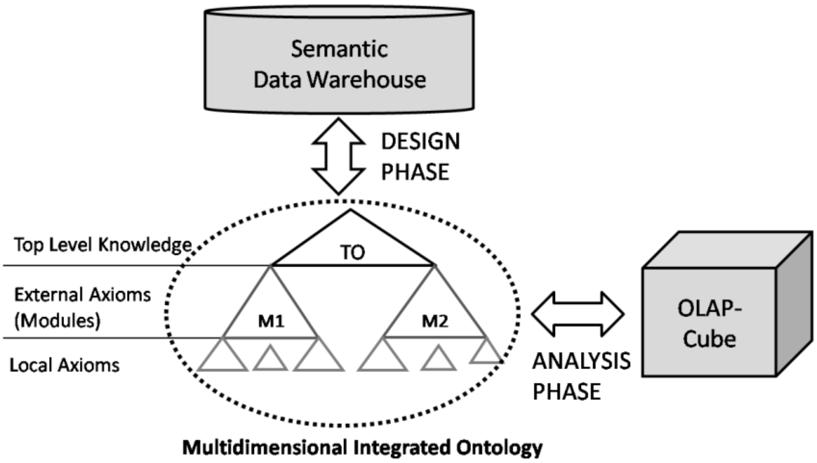

Figure 5. Analyzing an SDW through a MIO.

Figure 5 shows the intermediate role played by a MIO during the design and analysis of an SDW. On one hand, the MIO represents a consistent subset of the data from the SDW which covers the requirements stated by the analyst. On the other hand, this subset of data is used to build well-formed OLAP cubes for the multidimensional analysis.

Following the notation of Figure 4, the framework workflow has the following four phases:

Phase 1. *MIO Definition*: In this step, the analyst manually describes the topic of analysis, measures and dimensions that constitute the multidimensional conceptual schemata. In order to accomplish this task, the analyst applies a *Symbol Searcher*, which retrieves the symbols (concepts and properties) to be used in the analysis from the SDW domain ontologies.

Phase 2. *MIO Generation*: Once the analyst has defined the MIO, the *Module Extractor* tool automatically generates the corresponding ontology with the following elements:

1. A set of modules having the necessary knowledge to make inferences with the external symbols included in the MIO. For example, the application ontology of Figure 2 uses external symbols to define the diagnosis of a disease. These symbols come from the NCI ontology, which contains axioms that are necessary for reasoning.
2. A top-level ontology with the knowledge required to integrate the previous modules. The top-level ontology is the result of the union of the upper modules (UM) extracted from the used domain ontologies in the MIO. Additionally, a set of axioms derived from the ontology mappings are included to reconcile overlapping concepts.
3. The local axioms derived from the definitions given by the analyst when defining the MIO (Phase 1).

Phase 3. *MIO Validation*: To conclude the design phase, the resulting ontology is validated in the *Consistency Checker* tool in order to ensure that it will be able to generate the target cubes. If there is any inconsistency, the user is allowed to change axioms of the MIO so that a valid cube can be obtained.

Phase 4. *Analysis Phase*: During this phase, all the instances that will be used for generating the facts and dimensions for an OLAP cube, are retrieved by the *Instance Extractor*. Furthermore, there is a complex process called *OWLtoMDS* which must take into account the restrictions of the target OLAP tool (e.g. if it only allows strict hierarchies, level stratification, covering hierarchies, etc.) to transform the hierarchies of the MIO into suitable ones for analysis. Due to the inherent complexity of this task, it is out of the scope of this paper to provide a description of these transformations. However, we believe a formal method should be designed, which should take into account the analyst preferences regarding the definition of category levels, stratification and so on while making the process as easy and automatic as possible (Pedersen et al, 1999). Finally, facts are generated by applying some *Transformations* to the retrieved instances so that they conform to the multidimensional schema. In this step, ontology mappings can be required to transform instances that are non-compliant with the MIO. As a result, the final OLAP cube is generated. In the next subsections we will discuss the details of the design phase of the proposed methodology by means of a running example.

5.1. Phase 1: Defining the MIO

A MIO definition specifies a set of dimensions and measures that can be extracted from the Semantic Data Warehouse ontologies. The design process proposed here consists of five steps in which the analyst applies the available ontologies to design a new one with the elements needed for analysis task. The five steps are as follows: to select the topic of analysis, to specify the dimensions of analysis, to select the measures, to define potential roll-up relationships, and finally, to specify the instances to be analysed. We will develop the use case specified in Section 2 in order to illustrate the steps of the design process of the MIO. Remember that the objective of the use case is to analyse the efficacy of different drugs in the treatment of several types of inflammatory diseases, mainly rheumatic ones.

Step 1. In the first step, the topic of analysis is defined by selecting the concepts that are the focus of the analysis from the application ontologies. We denote by C^O a concept C taken from the ontology O. In our running example, the chosen concept is $\mathsf{Patient}^{\mathsf{Rheuma}}$.

Notice that Patient$^{\text{Rheuma}}$ represents all the patients defined in the Rheumatology application ontology.

Step 2. Next, the concepts that will be used in the dimensions of analysis must be specified (see Table 2). In this step, the local concepts of the categories in each dimension are first defined and then related to the external concepts coming from the ontologies used for the stored annotations. The following table shows the concepts selected for defining the dimensions included in the MIO specified for the analysis case of our running example.

Dim.	Description	Associated local concepts	Associated external concepts
D1	Diseases associated to body parts	{Disease$^{\text{LOCAL}}$, Anatomy$^{\text{LOCAL}}$}	{Disease$^{\text{LOCAL}}$ $\sqsubseteq$ Rheumatoid_Arthritis$^{\text{NCI}}$, Anatomy$^{\text{LOCAL}}$ $\sqsubseteq$ Anatomy_Kind$^{\text{NCI}}$ }
D2	Drugs used in treatments	{Drug$^{\text{LOCAL}}$}	{ Drug$^{\text{LOCAL}}$ $\sqsubseteq$ Drug$^{\text{UMLS}}$}
D3	Patient age	{Age$^{\text{LOCAL}}$, AgeGroup$^{\text{LOCAL}}$}	
D4	Patient sex	{Sex$^{\text{LOCAL}}$}	
D5	Biomarkers associated to tissues	{Biomarker$^{\text{LOCAL}}$, Tissue$^{\text{LOCAL}}$}	{ Biomarker$^{\text{LOCAL}}$ $\sqsubseteq$ AbsoluteMeasurement$^{\text{GALEN}}$, Biomarker$^{\text{LOCAL}}$ $\sqsubseteq$ Gene$^{\text{UMLS\_GENE}}$, Tissue$^{\text{LOCAL}}$ $\sqsubseteq$ Tissue$^{\text{GALEN}}$, Tissue$^{\text{LOCAL}}$ $\sqsubseteq$ Tissue$^{\text{NCI}}$ }
D6	Damage Index	{DamageIndex$^{\text{LOCAL}}$, DamageIndex_Group$^{\text{LOCAL}}$}	
D7	Follow-up (number of visit)	{NumberOfVisit$^{\text{LOCAL}}$}	

Table 2. Concepts associated to the ontology dimensions and external concepts they relate to.

In order to relate these local concepts to external ones, a set of axioms has been stated (see col. 4 of Table 2). For example, the axiom Disease$^{\text{LOCAL}}$ $\sqsubseteq$ Rheumatoid_Arthritis$^{\text{NCI}}$ states that the symbols used for the disease dimension will be the same as those used in the domain ontology NCI under the concept Rheumatoid_Arthritis. Then, it will be possible to do the same inferences over these symbols as over the original ontology. In other words, the semantics given by the NCI ontology is assumed for our Disease dimension.

As for dimension *D5*, the analyst wants to relate biomarkers (e.g. blood indicants and genes) to tissues. We have performed a review of the main biomedical ontologies searching for this kind of information and we have found GALEN to contain information about blood indicants and its relation to tissues (this relation is trivial since blood indicants measure blood cells, which are found in blood tissue). However, we have not found one or more ontologies that explicitly relate genes to specific cells or tissues. Thus, we have decided to define a tailored ontology that contains this information. Both the classification of genes and cells have been taken from UMLS. Then, we have manually established the corresponding relations based on the literature. We have named this ontology UMLS_GENES.

It is important to notice that in the application ontology of our example, there are no concepts associated with Age, so the dimension *D*3 must be derived from the data type property age. In this case, we have created the new concept Age whose instances will be derived from age range values. The concept AgeGroup is defined locally to account for the different patient age groups, for example: newborn, child, juvenile, adult and elderly people. The transformation of numerical values into Age instances is performed during the construction of the OLAP cube.

Step 3. The next step of the process consists of selecting the candidate measures coming from the data type properties existing in the application ontology. In our running use case, the DamageIndex could be a measure. The measure that counts the number of affected cases, like the other aggregation measures (e.g.: *sum*, *avg*, etc.), cannot be specified at this stage due to the DL expressivity limitations. This kind of measures will be defined and calculated during the analysis phase over the cube built from the MIO. As a consequence, measures are treated as *dimensions* in the MIO, like in (Pedersen et al., 2001).

Step 4. Roll-up relationships are the next elements to be defined. Local roll-up properties are represented as $R\_C_i\_C_j$, denoting that instances of the concept C_i will be rolled-up to instances of the concept C_j. As Table 2 shows, the local concepts $\text{Disease}^{\text{LOCAL}}$ and $\text{Anatomy}^{\text{LOCAL}}$ have been defined to represent the categories of dimension *D1*. Then, the roll-up relationship $\text{R\_Disease\_Anatomy}^{\text{LOCAL}}$ is created and relates both categories through the next local axiom:

$$\text{Disease}^{\text{LOCAL}} \sqsubseteq \exists \text{R\_Disease\_Anatomy}^{\text{LOCAL}}.\text{Anatomy}^{\text{LOCAL}}$$

which restricts the local concept Disease to roll up to an Anatomy concept. Analogous axioms are added for the rest of dimension categories. In Step 2 both local concepts have associated to external ones ($\text{Rheumatoid\_Arthritis}^{\text{NCI}}$ and $\text{Anatomy\_Kind}^{\text{NCI}}$ respectively). Therefore, the system will try to find an external roll-up relationship (e.g. path of subsequent concepts and properties) in external ontologies that connects both external concepts. In this case, the following path has been found:

$$\text{Rheumatoid\_Arthritis}^{\text{NCI}} / \text{Disease\_Has\_Associated\_Anatomic\_Site}^{\text{NCI}} / \text{Anatomy\_Kind}^{\text{NCI}}$$

Therefore, the following axiom associates the local roll-up property defined with the external roll-up property found in the ontology:

$$\text{Disease\_Has\_Associated\_Anatomic\_Site}^{\text{NCI}} \sqsubseteq \text{R\_Disease\_ Anatomy}^{\text{LOCAL}}$$

Table 3 shows the set of local roll-up relationships along with their corresponding external ones defined for each dimension of the running example.

Dim.	Local roll-up relationship	External roll-up relationship
		Axiom associating local and external roll-ups
D1	**R_Disease_ Anatomy$^{\text{LOCAL}}$**	Rheumatoid_Arthritis$^{\text{NCI}}$ / Disease_Has_Associated_Anatomic_Site$^{\text{NCI}}$ / Anatomy_Kind$^{\text{NCI}}$
		Disease_Has_Associated_Anatomic_Site$^{\text{NCI}}$ $\sqsubseteq$ R_Disease_ Anatomy$^{\text{LOCAL}}$
D3	**R_Age_AgeGroup$^{\text{LOCAL}}$**	
D5	**R_Biomarker_Tissue$^{\text{LOCAL}}$**	Gene$^{\text{UMLS\_GENE}}$ / Located_In$^{\text{UMLS\_GENE}}$ / Cell$^{\text{UMLS\_GENE}}{}_{m}^{\rightarrow NCI}$/ Anatomic_Structure_Is_Physical_Part_Of$^{\text{NCI}}$ / Tissue$^{\text{NCI}}$
		Located_In$^{\text{UMLS\_GENE}}$ $\circ$ Anatomic_Structure_Is_Physical_Part_Of$^{\text{NCI}}$ $\sqsubseteq$ R_Biomarker_Tissue$^{\text{LOCAL}}$
D5	**R_Biomarker_Tissue$^{\text{LOCAL}}$**	AbsoluteMeasurement$^{\text{GALEN}}$ / isCountConcentrationOf$^{\text{GALEN}}$ / Cell$^{\text{GALEN}}$ / isInSuspensionWithin$^{\text{GALEN}}$ / Tissue$^{\text{GALEN}}$
		isCountConcentrationOf$^{\text{GALEN}}$ $\circ$ isInSuspensionWithin$^{\text{GALEN}}$ $\sqsubseteq$ R_Biomarker_Tissue$^{\text{LOCAL}}$
D6	**R_DamageIndex_DamageIndexGroup$^{\text{LOCAL}}$**	

Table 3. Roll-up axioms defined for the MIO of the use case. We use the DL constructor ∘ to represent the role composition. Additionally, we use $s^{O1}{}_{m}^{\rightarrow O2}$ to denote a transformation of symbol *s* from ontology *O1* to *O2* by using a mapping *m*.

The local axioms that represent roll-up relationships are defined, when possible, composing roles (object properties) from the external ontologies. The external roll-up relationship found for R_Biomarker_Tissue$^{\text{LOCAL}}$ involves two different ontologies (UMLS_GENE and NCI). We have made use of mappings in order to relate cells of both ontologies.

Step 5. In the last step of the MIO design process, the instances to be analyzed are specified through a local concept that involves all the dimensions and measures previously defined:

$$\text{Patient}^{\text{LOCAL}} \equiv \exists \text{hasDim\_D1}^{\text{LOCAL}}.\ \text{Disease}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D2}^{\text{LOCAL}}.\text{Drug}^{\text{LOCAL}} \sqcap$$

$$\exists \text{hasDim\_D3}^{\text{LOCAL}}.\text{Age}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D4}^{\text{LOCAL}}.\text{Sex}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D5}^{\text{LOCAL}}.\text{Biomarkers}^{\text{LOCAL}} \sqcap$$

$$\exists \text{hasDim\_D6}^{\text{LOCAL}}.\text{DamageIndex}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D7}^{\text{LOCAL}}.\text{NumberOfVisit}^{\text{LOCAL}}$$

Additionally, a set of local axioms must be stated to relate dimension properties to external properties. Table 4 shows the axioms proposed for the running example. It is worth mentioning that *D5* (biomarkers) involves three different parts of the application ontology, namely: blood cell, factors and genes.

Dim.	Axioms associated to cube definition
D1	$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_diagnosis}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D1}^{\text{LOCAL}}$
D2	$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_Section}^{\text{Rheuma}} \circ \text{has\_therapy}^{\text{Rheuma}} \circ \text{has\_drug}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D2}^{\text{LOCAL}}$
D3	$\text{age}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D3}^{\text{LOCAL}}$
D4	$\text{sex}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D4}^{\text{LOCAL}}$
D5	$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_Section}^{\text{Rheuma}} \circ \text{measures\_indicant}^{\text{Rheuma}} \circ \text{has\_Blood\_Cell}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D5}^{\text{LOCAL}}$
D5	$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_Section}^{\text{Rheuma}} \circ \text{measures\_indicant}^{\text{Rheuma}} \circ \text{has\_Blood\_Factor}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D5}^{\text{LOCAL}}$
D5	$\text{has\_Profile}^{\text{Rheuma}} \circ \text{related\_gene}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D5}^{\text{LOCAL}}$
D6	$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_Section}^{\text{Rheuma}} \circ \text{DamageIndex}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D6}^{\text{LOCAL}}$
D7	$\text{has\_Report}^{\text{Rheuma}} \circ \text{dateOfVisit}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D7}^{\text{LOCAL}}$

Table 4. Axioms associated with the intended facts of the target cube.

5.2 Phase 2: MIO generation

After completing the design of the MIO, the analyst has defined the topic of the analysis, the external concepts associated with dimensions, the roll-up relationships between dimension concepts and their links to external properties. Next, the system will automatically generate the MIO. This will consist of the following three elements:

$$MIO = \bigcup_{\forall Di} LocalAxioms(Di) \bigcup TopicAxioms \bigcup ExternalAxioms$$

The set of local axioms for each dimension D_i, denoted *LocalAxioms*(D_i), will be built as the union of all the relevant specifications of the design process. For example, for the dimension *D1* we have:

LocalAxioms(*D1*)={

$\text{Disease}^{\text{LOCAL}} \sqsubseteq \text{Rheumatoid\_Arthritis}^{\text{NCI}}$,

$\text{Disease}^{\text{LOCAL}} \sqsubseteq \exists \text{R\_Disease\_Anatomy}^{\text{LOCAL}}.\text{Anatomy}^{\text{LOCAL}}$,

$\text{Anatomy}^{\text{LOCAL}} \sqsubseteq \text{Anatomy\_Kind}^{\text{NCI}}$,

$\text{Disease\_Has\_Associated\_Anatomic\_Site}^{\text{NCI}} \sqsubseteq \text{R\_Disease\_ Anatomy}^{\text{LOCAL}}$,

$\text{has\_Report}^{\text{Rheuma}} \circ \text{has\_diagnosis}^{\text{Rheuma}} \sqsubseteq \text{hasDim\_D1}^{\text{LOCAL}}$

}

The *TopicAxioms* will also be built from the specifications previously made for the topic of analysis and the measures. In our example, we will have:

TopicAxioms = {

$\text{Patient}^{\text{LOCAL}} \equiv \exists \text{hasDim\_D1}^{\text{LOCAL}}.\ \text{Disease}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D2}^{\text{LOCAL}}.\text{Drug}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D3}^{\text{LOCAL}}.\text{Age}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D4}^{\text{LOCAL}}.\text{Sex}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D5}^{\text{LOCAL}}.\text{Biomarkers}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D6}^{\text{LOCAL}}.\text{DamageIndex}^{\text{LOCAL}} \sqcap \exists \text{hasDim\_D7}^{\text{LOCAL}}.\text{NumberOfVisit}^{\text{LOCAL}}$ }

Therefore, at this stage it only remains to generate the *ExternalAxioms* element. The following section deals with this issue, and the subsequent section explains how to validate the resulting ontology.

5.3. Bringing external knowledge to the MIO

Concepts that will be used in the different dimensions are defined locally, but the user defines them in terms of the concepts located in external ontologies. Thus, a MIO consists of all the local axioms asserted by the user plus external knowledge that can affect the symbols of the MIO. It is desirable to integrate this external knowledge because of three reasons:

1. Semantic annotations made with symbols from domain ontologies can imply definitions and relationships that are implicit. Thus, by enriching the MIO with new hierarchical dimensions relying on the relationships provided by domain ontologies, we can discover implicit knowledge. In other words, bringing in the knowledge related to the symbols of the warehouse semantic annotations, allows us to infer implicit fact-dimension relationships useful for analysis.

2. Given that a MIO contains a set of external axioms that provides a consistent and simplified version of the original ontologies focused on a topic of analysis, it constitutes a piece of knowledge that can be reused. For example, this MIO can be a good starting point to guide users in the definition of a multidimensional cube for analysis purposes. There exists some preliminary work in this line that could benefit from MIOs (e.g. Romero & Abelló 2007).

3. A MIO is a new consistent ontology that derives from the SDW ontologies. This means that it can contain new concepts and roles that must be satisfiable with respect to the semantics of the original ontologies. We assume that the original ontologies are already consistent, and therefore satisfiability must be checked only for the MIO local concepts. In this way, although large MIOs can be defined by re-using existing knowledge, the cost of checking it for consistency is limited to the new concepts introduced by the analyst.

The construction of the MIO with external knowledge coming from the domain and application ontologies is carried out by using both the query language OntoPath (Jimenez-Ruiz et al., 2007) and some module extraction approaches recently proposed in (Jimenez-Ruiz et al., 2008).

OntoPath is a novel retrieval language for specifying and retrieving relevant ontology fragments. This language is intended to extract customized stand-alone ontologies from very large, general-purpose ones. In a typical OntoPath query, the desired detail level in the concept taxonomies as well as the properties between concepts that are required by the target applications are easily specified. The syntax and aims of OntoPath resemble XPath's in the sense that they are simple and they are designed to be included in other XML-based applications (e.g. transformations sheets, semantic annotation of web services, etc.). In our approach for building the MIO, OntoPath is used to retrieve the different dimension hierarchies along with the corresponding roll-up properties from the domain ontologies used to annotate patients. The retrieval of these ontology fragments is based on the analysis dimensions proposed by the analyst. Following the running example, the following queries would be run in order to extract the dimension hierarchies:

D1 → Rheumatoid_Arthritis^{NCI}/ Disease_Has_Associated_Anatomic_Site^{NCI}/ Anatomy_Kind^{NCI}

D2 → Drug^{UMLS}

D5 → Gene$^{UMLS\_GENE}$ / Located_In$^{UMLS\_GENE}$ / Cell$^{UMLS\_GENE}\ {}_{m}^{\rightarrow NCI}$ /
Anatomic_Structure_Is_Physical_Part_Of^{NCI} /Tissue^{NCI}

D5 → AbsoluteMeasurement^{GALEN} / isCountConcentrationOf^{GALEN} / Cell ^{GALEN} /
isInSuspensionWithin^{GALEN} /Tissue^{GALEN}

As it can be observed, through simple path queries of subsequent concepts and properties, we obtain the fragments corresponding to the different dimension hierarchies. Notice that we make use of mappings in D5 in order to connect overlapping concepts in different ontologies. OntoPath is also used for extracting the part of the application ontology schema relevant for analysis purposes; the concepts and properties that define the facts of analysis. In our example, the OntoPath query shown in Figure 6 is evaluated to determine the relevant elements of the application ontology involved in the analysis task.

```
Patient$^{Rheuma}$
      [age$^{Rheuma}$ ]
      [sex$^{Rheuma}$]
      [has_Profile / * / related_gene / * ]
       [ has_Report$^{Rheuma}$ / *
            [ has_diagnosis$^{Rheuma}$ / *]
            [dateOfVisit$^{Rheuma}$ / *  ]
            [ has_Section$^{Rheuma}$ / *
               [DamageIndex$^{Rheuma}$ ]
               [has_therapy$^{Rheuma}$ / *
                  [ has_drug$^{Rheuma}$ / * ]
               [measures_indicant$^{Rheuma}$ / *
                  [ has_Blood_Cell$^{Rheuma}$ / * ]
                  [ has_Blood_Factor$^{Rheuma}$ /* ]
               ]
            ]
      ]
```

Figure 6. OntoPath query for the application ontology of the use case. In OntoPath, the symbol "*" denotes any concept, and nested expressions (e.g. tree branches) are in brackets like in XPath.

Moreover, we use a logic-based approach of modular reuse of ontologies to extract the upper knowledge of all the external symbols that appear in the MIO. This modular approach is *safe*, since the meaning of the imported symbols is not changed, and *economic*, since only the module relevant for a given set of symbols (called signature) is imported. They also guarantee that no entailments are lost compared to the import of the whole ontology. We particularly extract Upper Modules (UM), which are based on ⊥-locality and are suitable for refinement. That is, we extract the upper knowledge of all the external symbols of the MIO.

In our use case, we group the external symbols according to the external ontologies they are pointing to. Then, a module containing the upper knowledge of each signature is extracted. The external signatures for our use case are the following ones:

Sig$^{\text{Rheuma}}$={Patient$^{\text{Rheuma}}$}

Sig$^{\text{NCI}}$ = { Disease_Has_Associated_Anatomic_Site$^{\text{NCI}}$, Cell$^{\text{NCI}}$, Tissue$^{\text{NCI}}$, Anatomy_Kind$^{\text{NCI}}$, Rheumatoid_Arthritis$^{\text{NCI}}$, Anatomic_Structure_Is_Physical_Part_Of$^{\text{NCI}}$}

Sig$^{\text{UMLS\_GENE}}$ = { Gene$^{\text{UMLS\_GENE}}$, Located_In$^{\text{UMLS\_GENE}}$, Cell$^{\text{UMLS\_GENE}}$ }

Sig$^{\text{GALEN}}$ = {AbsoluteMeasurement$^{\text{GALEN}}$, isCountConcentrationOf$^{\text{GALEN}}$, Cell$^{\text{GALEN}}$, isInSuspensionWithin$^{\text{GALEN}}$, Tissue$^{\text{GALEN}}$}

The top knowledge ontology is composed by the union of the upper modules extracted plus some additional axioms derived from the stored mappings that allow merging the upper knowledge of overlapping concepts. Mappings are stored in the data warehouse as 7-tuples $\langle id, s_1, s_2, O_1, O_2, \mathrm{R}, \phi\rangle$, where s_1 and s_s are symbols from ontologies O_1 and O_2 respectively, ϕ is a confidence value and R is the mapping relationship between these symbols, namely: equivalent (≡), subsumption (⊑) and disjointness (⊥). For each pair of top knowledge concepts s_1, s_2 for which a mapping is recorded, we add the corresponding axiom according to the mapping relationship: ***equivalentTo***(e_1, e_2) for (≡), ***subClassOf***(e_1, e_2) for (⊑) and ***disjoint***(e_1, e_2) for (⊥).

As an example of the type of knowledge extracted with the previous approaches, in Figure 7, we show a fragment of the axioms extracted with the UM approach and the OntoPath tool about the concept Rheumatoid_Arthritis under Disease$^{\text{NCI}}$.

Upper Module	Ontopath-based Module
Rheumatoid_Arthritis ⊑ Autoimmune_Disease	Rheumatoid_Arthritis ⊑ ∃ Disease_Has_Associated_Anatomic_Site. Connective_and_Soft_Tissue
Autoimmune_Disease ⊑ Immune_System_Disorder	Stills_Disease ⊑ Rheumatoid_Arthritis
Immune_System_Disorder ⊑ Non-Neoplastic_Disorder_by_Special_Category	Oligoarticular_Stills_Disease ⊑ Stills_Disease
Non-Neoplastic_Disorder_by_Special_Category ⊑ Non-Neoplastic_Disorder	Synovial_Membrane ⊑ Connective_and_Soft_Tissue

Figure 7. External knowledge involved in Rheumatoid_Arthritis.

Finally, in the current implementation, the MIO is composed by a set of OWL files connected through “import” statements gathering together the local axioms, topic axioms and external axioms.

5.4. Phase 3: MIO Validation

The MIOs are validated at two levels: schema and instance. At the former level, we check that the generated ontology is consistent with respect to all the asserted axioms: local and external ones. If the ontology is not consistent, then we cannot generate a valid OLAP cube for it and the ontology should be fixed. For this purpose, it is necessary to detect invalid dimensions that constitute potentially not valid cubes. At the second level, once the multidimensional ontology is validated, it must be populated with instances from the data warehouse. The issues of this process will be explained in the following section.

A MIO is a formal ontology in which all the knowledge has been included in order to perform the appropriate inferences and queries. This knowledge can also be used for checking certain properties and in this way, ensuring that not-valid final cubes will not result. In (Hurtado & Mendelzon, 2002) a set of structural constraints are applied to check some interesting properties of heterogeneous dimensions. These properties could be checked over the MIO ontology to indicate to the analyst that potential problems could arise in the final OLAP-based cube. Unfortunately, some of these properties can only be checked once the cube is formed (e.g. summarizability) as they depend on the specific dimension values and aggregation functions defined for the target cube. The set of properties that we can check in the multidimensional ontology are the following:

- **Disjointness.** The member set of two categories belonging to the same dimension must be disjoint. Notice that with this constraint **Stratification** is also achieved, as any instance of a category can only roll up to an upper category instance.

- **Category satisfiability.** Another inference problem stated in (Hurtado & Mendelzon, 2002) is the satisfiability of a category in a dimension schema. Basically, this means that at least there exists an instance of the schema in which the member set of the category is not empty. This is equivalent to the problem of checking the satisfiability of the dimension classes with respect to the axioms of the MIO.

- **Shortcut free.** This property is also known as "non-covering" in the OLAP literature (Pedersen et al. 2001). A shortcut occurs when a fact can be rolled up from a category C_i to another C_j without passing through an intermediate category C_x that connects both of them. This is true when the MIO contains the roles $R\_C_i\_C_x$, $R\_C_x\_C_j$ and $R\_C_i\_C_{j.}$ In other words, the graph formed by the concepts (nodes) and the set of roll-up relationships (edges) of each dimension, must not contain redundant edges. Moreover, ensuring that this graph is connected, and assuming that every instance can roll up to an instance of the concept Thing ($\top$), we also ensure the **Up-Connectivity** property.

- **Orthogonality.** This is the property of having a set of dimensions without dependency relationships. Dimension dependencies produce sparse cubes, as many combinations of dimension values are disallowed. Having dependent dimensions is considered a bad conceptual design (Abelló, 2002), although sometimes this is desired by the designer. In our case, we have to check when two categories of different dimensions are somehow related. Thus, first it must be ensured that the concepts of two different dimensions are all disjoint, and second that there does not exist any chain of properties relating two concepts of different dimensions (Romero & Abelló, 2007).

- **Summarizability** (Lenz & Shoshani, 1997)**.** The only way to achieve this property is by ensuring all the previous properties plus the functionality of all the roll-up properties. As it is difficult to ensure functionality from the original ontologies, this property will be checked over the final generated facts and dimensions. Notice that some multidimensional models (e.g. Pedersen et al., 2001) are able to deal with many-to-many relationships. This means that forcing functionality will depend on the features of the target multidimensional model.

In the running example, *disjointness* is achieved by asserting the following axiom:

$$\text{alldisjoint(Disease}^{\text{LOCAL}}\text{,Anatomy}^{\text{LOCAL}}\text{, Biomarker}^{\text{LOCAL}}\text{,Tissue}^{\text{LOCAL}}\text{, Age}^{\text{LOCAL}}\text{, AgeGroup}^{\text{LOCAL}}\text{, ,}$$
$$\text{Sex}^{\text{LOCAL}}\text{, Drug}^{\text{LOCAL}}\text{, Follow-up}^{\text{LOCAL}}\text{, DamageIndex}^{\text{LOCAL}}\text{, DamageIndexGroup}^{\text{LOCAL}}\text{)}$$

The resulting MIO is satisfiable and shortcut free. However, it can be demonstrated by using the axioms of the MIO that dimensions *D1* and *D5* are dependent, and therefore not completely orthogonal. For example, the following axioms show a dependency between the disease RA and the biomarker IL6:

- **Rheumatoid_Arthritis** ⊑ Disease ⊓ ∃ Disease_Has_Associated_Anatomic_Site. Connective_and_Soft_Tissue
- Connective_and_Soft_Tissue ⊑ Tissue
- **IL6** ⊑ Biomarker ⊓ ∃ Expressed_In_Cell.Synovial_Cell
- Synovial_Cell ⊑ Cell ⊓ ∃Anatomic_Structure_Is_Physical_Part_Of.Synovial_Membrane
- Synovial_Membrane ⊑ Connective_and_Soft_Tissue

Here, we can conclude that both concepts are related somehow with Connective_and_Soft_Tissue. Similarly, we can find some dependency between RA and blood sample biomarkers as RA is an autoimmune disease that mainly affect to macrophage cells in the blood. Indeed, the original definition of biomarker is that it provides clues to diagnose a disease, thus the strong dependency between both concepts.

5.5. Phase 4: OLAP-based analysis

Before building the target OLAP-based cube, the MIO must be properly populated with the instances from the Semantic Data Warehouse that satisfy both the MIO and the set of specific roll-up relationships between them. This process consists of two phases: (1) the retrieval of ontological instances from the data warehouse, and (2) the transformation of the instances with an appropriate granularity for the OLAP cube. Additionally, the cube dimensions and their possible categories must be also built from the MIO concepts and roles. Subsequent sections describe these aspects with detail.

5.5.1. Instance Retrieval

Application ontology instances are stored in a *RDF triple store* like *3store* (Harris and Gibbins, 2003) as shown in Table 1. The objective of this phase is to retrieve the appropriate instances that can populate the MIO. In order to accomplish this task we have considered two approaches. The first one seems the most straightforward and consists of using the triple store reasoning capabilities in order to extract all the required instances. A triple store such as *3store* claims to support efficient processing of RDQL queries and RDF(S) entailments (RDF(S) entailments are not implemented in SparQL, the successor of RDQL). Therefore, it is trivial to translate the OntoPath query of Figure 6 into a set of RDQL queries that use the reasoning capabilities provided to extract the instances. However, some experiments have demonstrated that this kind of triple store is not scalable when dealing with RDF(S) entailments over ontologies of considerable size (e.g. a few thousand concepts and properties). Thus, a more long term solution must be devised.

The second approach consists of leaving the RDF(S) entailments to OntoPath and use the triple store with the inference capabilities off. The OntoPath query of Figure 6 used for extracting the part of the AO schema relevant for analysis purposes is the one that dictates the instances to be retrieved from the SDW. The result of the above mentioned query is twofold. On one hand, OntoPath returns the sub-ontology that matches the query in the form of OWL primitives. This feature is useful when extracting the AO schema as well as the different fragments corresponding to the dimension hierarchies from domain ontologies in order to build the MIO. On the other hand, OntoPath can present the result of a query as a result set consisting of all the different sub-graphs of an ontology that match the query (RDFS entailments). Then, every OntoPath sub-graph from the result set can be translated into an appropriate RDF query language, such as SparQL. That is, every possible sub-graph returned by OntoPath corresponds to a SparQL query without RDFS entailments. Figure 8 shows part of the OntoPath query for our use case, the OntoPath result set and the translation of each sub-graph into SparQL.

<table>
<tr><th colspan="2">OntoPath Query</th></tr>
<tr><td colspan="2">Patient$^{\text{Rheuma}}$ [has_Report$^{\text{Rheuma}}$ / * / has_Section$^{\text{Rheuma}}$ / * / has_therapy$^{\text{Rheuma}}$ / * / has_drug$^{\text{Rheuma}}$ / *]</td></tr>
<tr><th>OntoPath Result Set (sub-graphs matching)</th><th>SparQL Translation</th></tr>
<tr><td>Patient$^{\text{Rheuma}}$ [has_Report$^{\text{Rheuma}}$ / Rheumatology_Report / has_Section$^{\text{Rheuma}}$ / Treatment / has_therapy$^{\text{Rheuma}}$ / Drug_Therapy / has_drug$^{\text{Rheuma}}$ / Drug$^{\text{UMLS}}$]</td><td>SELECT *
WHERE {
 ?person type Patient .
 ?person has_Report ?report .
 ?report type Rheumatology_Report .
 ?report has_Section ?section .
 ?section type Treatment .
 ?section has_therapy ?t .
 ?t type Drug_Therapy .
 ?t has_drug ?drug .
 ?drug type DrugUMLS
}</td></tr>
<tr><td>Patient$^{\text{Rheuma}}$ [has_Report$^{\text{Rheuma}}$ / Rheumatology_Report / has_Section$^{\text{Rheuma}}$ / Treatment / has_therapy$^{\text{Rheuma}}$ / Joint_Injections / has_drug$^{\text{Rheuma}}$ / Drug$^{\text{UMLS}}$]</td><td>SELECT *
WHERE {
 ?person type Patient .
 ?person has_Report ?report .
 ?report type Rheumatology_Report .
 ?report has_Section ?section .
 ?section type Treatment .
 ?section has_therapy ?t .
 ?t type Joint_Injections .
 ?t has_drug ?drug .
 ?drug type DrugUMLS
}</td></tr>
</table>

Figure 8. Translating from OntoPath sub-graphs into SparQL. Notice the OntoPath query results in two sub-graphs since the range of *has_therapy*$^{\text{Rheuma}}$ matches *Drug_Therapy* and also *Joint_Injections*, which is a subclass of *Drug_Therapy*.

5.5.2. Instance transformations

There are two kinds of transformations that must be applied to the retrieved instances and values in order to obtain consistent MIO instances, namely: 1) to convert data type values (or data type property ranges) into new instances and, 2) to change instance identifiers and instance types according to the existing mappings.

The first kind of transformation is applied when a roll-up property is required over values instead of instances. For example, to roll up the feature hasAge into ageGroup we first need to convert ages (integer numbers) into instances, for example the value 32 is converted into the instance Age_32. This instance belongs to the class Age^{LOCAL} which has been defined in the MIO. Now, we can assert that Age_32 rolls up to the instance adult through the role R_Age_AgeGroup.

The second kind of transformations allows instances coming from different application ontologies to be expressed in the same terms within the MIO. This is performed by applying the existing mappings between the domain ontologies. For example, in our use case we have adopted NCI to represent disease concepts. If we want to include instances from an application ontology that uses GALEN for representing diseases, then we need to translate their instances to NCI terminology. This means to change their names as well as their types to NCI vocabulary.

Notice that mapping-based transformations can produce both incomplete and imprecise facts. Incomplete facts can be generated if the class of an instance has no (direct or inferred) mapping associated to the target ontology. Imprecise facts are generated when the mapping is inherited (i.e. it occurs for some super-class of the instance's class), and therefore the instance must be expressed with a broader concept.

Another required transformation for instances consists of changing the detail level at which they are expressed in the ontologies. For example, in the application ontology shown in Figure 2, all the instances related to drugs are borrowed from the domain ontology UMLS, but their type within the application ontology will be always Drug. This is because when the clinician is prescribing a drug to the patient, she is not concerned with the whole taxonomy in which the drug is placed but just with the drug's name. However, when analyzing patient data, the UMLS taxonomy for drugs is necessary to define dimension *D*2, and therefore the instances must have associated its actual type. For example, in Table 5, the instance Infliximab will change its type from Drug^{Rheuma} to AntiRheumaticAgent^{UMLS}.

Considering our use case, Table 5 shows a subset of the instances that populate the local concept Patient^{LOCAL}. In this case, the dimension *D3* has been generated by transforming the values of the data type property hasAge of the Rheumatology application ontology. Instances in dimensions *D1*, *D2* and *D5* have changed its type to that of the domain ontologies from which they are taken.

ID	*D1*	*D2*	*D3*	*D4*	*D5*	*D6*	*D7*
8787u	RA1	Infliximab	Age32	Male	Neutrophil↑	12	1
8991u	JIA1	Etacernept	Age15	Male	RF-	7	1
8991u	JIA1	Etacernept	Age15	Male	CProtein+	7	3
8882u	RA2	Naproxen	Age27	Female	HLA+	14	1
8882u	RA2	Naproxen	Age27	Female	HLA-	1	2
9912u	SD1	Methotrexate	Age34	Male	ESR↓	12	1

Table 5. Example of instances that populate the concept Patient^{LOCAL} in the MIO of the proposed use case. For biomarker instances (*D5*), we use the symbols + /– to denote presence/absence and ↑/↓ for high/low levels.

5.5.3. Generating cube dimensions

During the generation of the final analysis cube, the symbols of the MIO are interpreted as elements of the target multidimensional data model. Thus, concepts, properties and instances of the MIO will be interpreted as dimensions, categories, members, attributes and facts of the multidimensional model. Depending on the restrictions of the target multidimensional model, it can be necessary to transform some of the MIO symbols with the purpose of obtaining the proper interpretation. Moreover, many symbols of the ontology could be interpreted in different ways, resulting in very different cubes.

A dimension concept (e.g. *Disease*) is usually interpreted as a dimension category of the multidimensional data model. However, the members of these categories can be either the instances or the subclasses of the dimension concept. In the second case, as subclasses can be also hierarchically organised, they can produce further categories in the dimension. Figure 9 shows examples of these two interpretations. The members of the category Anatomy are the different anatomical instances (e.g. different body parts of each patient), whereas the members of the category Disease are the names of the sub-classes of Disease. Notice that two sub-categories are defined due to the hierarchical relationships between these sub-classes.

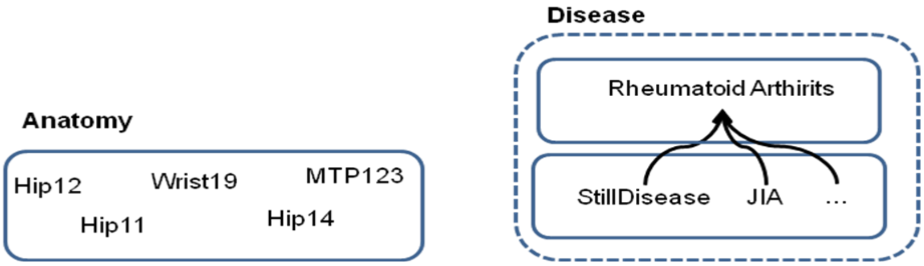

Figure 9. Two different interpretations for defining a dimension category.

Concerning the cube roll-up relationships between dimension categories, we also have different interpretations depending on the interpretation adopted for the involved categories. Thus, we have three possible interpretations, namely:

1. If both categories have instance members, then $R\_C_i\_C_j$ is interpreted at instance level too, and therefore each asserted triple (i_1, r, i_2) associated to $R\_C_i\_C_j$ defines a roll-up relation $RU(i_1, i_2)$.

2. If the lower category contains instance members and the upper one contains class names, then we interpret $R\_C_i\_C_j$ as before, but the roll-up relation is set to $RU(i_1,C_x)$, with $C_x \in Type(i_2)$ and $C_x \sqsubseteq C_j$.

3. If the related categories Ci and Cj contain class names, and they are connected with a roll-up role $R\_C_i\_C_j$, then we have two possible situations:

   - If there are no asserted instances associated to $R\_C_i\_C_j$, for each $R \sqsubseteq R\_C_i\_C_j$ such that $C'_i \in$domain(R) and $C'_j \in$range(R), a roll-up relation RU(C'_i,C'_j) is set.

   - Otherwise, the asserted triples (i_1, r, i_2) associated to $R\_C_i\_C_j$ defines a roll-up relation $RU(C_x, C_y)$ where $C_x \in Type(i_1)$ and $C_x \sqsubseteq C_i$ and $C_y \in Type(i_2)$ and $C_y \sqsubseteq C_j$.

It is worth mentioning that the selection of the interpretation is done by the analyst. Figure 10 shows examples of these three interpretations for some categories defined in the use case.

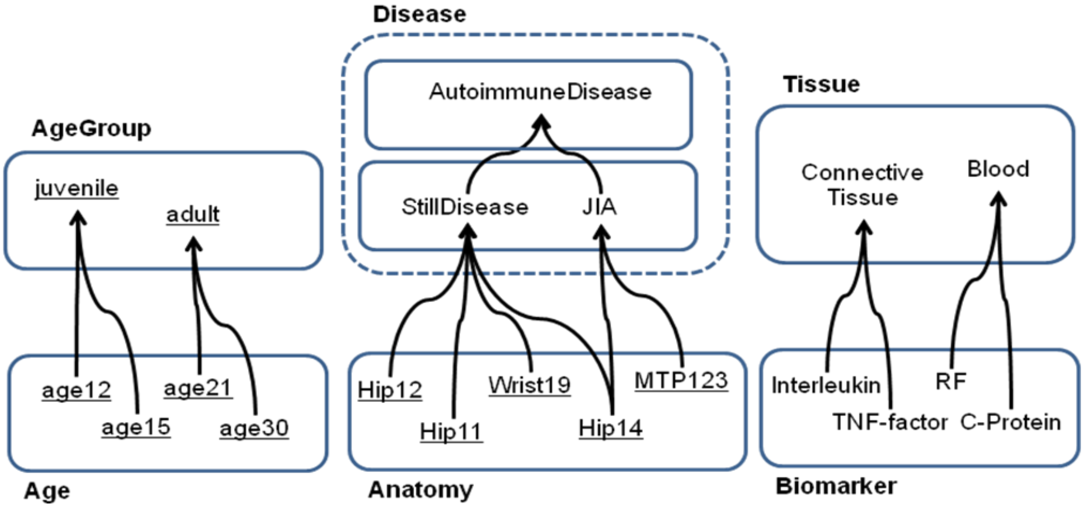

Figure 10. Different interpretations for roll-up relationships: instance-instance, instance-class and class-class roll-ups

Another relevant aspect to take into consideration when building roll-up hierarchies is the multiplicity between related categories. Ideally, each roll-up relationship should have a predominant multiplicity of many-to-one in order to properly aggregate data. In our use case, the role R_Disease_Anatomy however has a one-to-many predominant multiplicity, which means that it is not useful for aggregating data in the resulting cube. In order to include Anatomical information in the cube, we can either use the inverse role R_Anatomy_Disease or include *Anatomy* data in some attribute of the *Disease* members. The former solution is not valid in our use case as in the application ontology *Anatomy* concepts (e.g. SynovialJoint) and *Disease* concepts are not related to each other and therefore we cannot state reliable roll-up relations. In the second solution, we can only use anatomical data to restrict the diseases that the clinician wants to analyze. Finally, it is worth mentioning that *Disease* and *Anatomy* cannot be defined as two different dimensions because they are dependent on each other.

In order to complete the cube definition, additional member attributes can be taken from any of the properties associated to the MIO concepts that do not participate in the roll-up relationships.

The whole translation process from MIO to the target cube is a very complex task that will determine the possible analysis tasks to be performed through OLAP operations. As a consequence, this process deserves more attention in the future work in order to automate it as much as possible. A good starting point is the methods presented in (Pedersen et al., 1999).

6. Implementation Issues

Currently we have partially implemented the proposed framework for SDWs. In this section we describe the main issues we have addressed during this preliminary implementation.

In our first approach we have adopted the tried-and-tested “data warehousing” approach. Here, all source data is first extracted from the data sources (in our case both external, web-based sources and internal sources). Then, the data is transformed and various validation checks are performed. Some checks are completed before transformations are performed, and some after transformations (e.g., into a dimension) are performed, as described in Section 5. In order for the data to comply with the constraints, some *data cleansing* will be performed, e.g., new dimension members may be added in order to balance the hierarchy to achieve summarizability. Finally, the transformed data is stored in the SDW database. Because of the complex RDF-based structure of the ontologies, we have chosen an *RDF triplestore*, specifically 3store (Harris and Gibbins, 2003). Although 3store provides a limited form of logical reasoning based on the RDFS *subClassOf* hierarchies, it does not scale well. The reason is that it makes explicit all the entailments of the ontology. In this way, we have used 3store only for storing large sets of instances generated by the application ontologies, assuming that these ontologies do not contain large concept hierarchies and therefore do not require large sets of entailments.

Regarding the domain ontologies, the SDW must also provide the storage and querying mechanisms for them. Currently, there are a few approaches to store and query large OWL ontologies (Lu et al., 2007, Roldán-García et al., 2008). The main difference between these approaches and triplestores is that OWL stores must allow entailments with the same expressivity of the stored ontologies, which goes beyond the hierarchies defined in RDFS. Unfortunately, current OWL stores are not able to handle very large expressive ontologies, nor does current reasoners support secondary storage.

In our current implementation we have used both OntoPath and a series of labelling-based indexes specially designed to handle very large OWL-based ontologies (Nebot and Berlanga, 2008). These indexes allow the fast retrieval of sub-graphs and the fast construction of upper modules as those required by our methodology. It is worth mentioning that with these indexes we are able to check if one concept subsumes another by simply comparing two intervals. We have evaluated these indexes over the UMLS meta-thesaurus, which contains 1.5 million concepts and 13 million relationships. By using OntoPath indexes, we are able to build upper modules for signatures of hundreds of concepts in a few minutes. In this way, we achieve the scalability of the system by efficiently building customized modules, which can be handled by current reasoners.

Following the running example, in Table 6 we show some statistics about the different fragments extracted from external domain ontologies in order to enrich the dimension hierarchies. As it can be seen, the relative size of the fragments compared to the whole ontologies is drastically reduced, which shows the scalability of the MIOs used for analysis purposes. Similarly, Table 7 shows statistics about the top knowledge ontology, which is also part of the MIO. The top knowledge ontology is composed by the union of the upper modules extracted plus some additional axioms derived from the stored mappings that allow merging the upper knowledge of overlapping concepts. Once more scalability is assured since the size of the top knowledge is insignificant compared to the size of the original ontologies.

	# classes	**# properties**	**# subclass ax.**	**relative size %**
D1 (NCI)	65	1	65	0.24 %
D2 (UMLS)	699	0	1526	0.046 %
D5 (GALEN)	58	2	75	1.97 %
D5 (GO/NCI)	114	1	114	0.00027%

Table 6. Statistics about fragments extracted for dimension hierarchies.

	# classes	# properties	# subclass axioms	# total axioms	relative size %
D1 *(NCI)*	22	2	24	36	0.19 %
D2 *(UMLS)*	0	0	0	0	-
D5 *(GALEN)*	34	23	34	67	2.8 %
D5 *(UMLS/NCI)*	46	1	92	92	0.00011%
(RHEUMA)	1	0	1	1	-

Table 7. Statistics about top knowledge extracted from every ontology.

Concerning the ontology mappings, despite the large number of semi-automatic approaches that exist to generate them (see surveys presented in (Choi et al., 2006, Euzenat, 2007), current precision results are not good enough to make the automatic transformations proposed in this paper reliable. Moreover, most ontology matchers can only handle small ontologies (Hu et al., 2008), which limit their usefulness in our scenario. Fortunately, in our application scenario about Biomedicine, there exists a great interest in integrating existing knowledge resources. As a result, most ontologies are being annotated with UMLS terms and other standard vocabularies (e.g. NCI), which notably eases the mapping problem. Our preliminary experiments by using these vocabularies to link domain ontologies are promising.

7. Conclusions and Future Work

In this paper we have set the bases for the multidimensional analysis of Semantic Web data in a data warehouse. We have reviewed the work that combines data warehouse and semantic web technologies. From this review we conclude that XML-related technologies are becoming mature enough to enable the construction of semi-structured web data repositories. We have also highlighted the promising usage of the Semantic Web languages to integrate distributed data warehouses and to describe and automate the ETL process of a data warehouse. Regarding the analysis of semantically annotated data, the existing alternatives are only valid for single and small ontologies. Unfortunately, many real applications imply several large inter-linked ontologies.

As a solution, we have defined the Semantic Warehouse as an XML repository of ontologies and semantically annotated data of a particular application domain; and we have proposed a new framework to design conceptual multidimensional models starting from a set of application and domain ontologies. Our approach has a number of advantages. For example, the users can easily state facts and dimensions of analysis by selecting the relevant concepts from the ontologies. The methodology's underlying multidimensional model is very simple, only facts, measures, dimensions, categories and roll-up relationships need to be identified. This will allow us to implement the model in almost any existing multidimensional database by performing the proper transformations. Regarding the scalability of the approach, we are able to manage large-sized ontologies by selecting fragments representing semantically complete knowledge modules.

Modeling diagrams such as those proposed in (Abelló et al., 2006; Franconi & Ng, 2000) can be very helpful to guide users when defining a MIO. As future work, we plan to study how they can be coupled with ontology editors and reasoners to facilitate the creation of MIOs. Another interesting research line is to define appropriated indexing schemes for SDWs that enable the interaction of reasoners with OLAP tools.

Finally, we consider that addressing the temporal aspects of the semantic annotations, and the incremental consistency checking and reasoning with our MIO-based approach are also very attractive challenges.

In the future work we plan to carry out a deeper study of alternative implementations of SDWs. The main drawbacks of the current implementation include that the data may become outdated due to sources updates and that the extraction and validation process takes a long time to perform. A problematic issue which is particular to SDWs is that especially external data may have such a bad quality that the validation checks may disallow their integration in the materialized data warehouse, even if some parts of the data have sufficient quality. In this way, the options are either to allow bad data quality or to refuse some data to be admitted into the SDW.

An alternative to the materialized approach consists of a virtual implementation. That is, the SDW only exists as a collection of metadata, pointing to the underlying (external and internal) data sources. The actual extraction of data from the sources is not done until query time. This also means that the validation and other constraint checks will have to be done at query time. Here, the main difference from the materialized implementation is that only the data items and ontology parts directly related to the specific query being executed are extracted, transformed, and validated. This approach is quite similar to the virtual OLAP-XML integration engine (Pedersen et al., 2002). During query processing, a triplestore can be used for intermediate storage and processing (validation inference, etc.). Again, it will in the long term be more optimal to develop a dedicated query engine for this particular scenario. Because of the smaller data volumes, both a triplestore-based and a dedicated solution will be able to perform almost all processing in main memory. The advantages include that data is always up-to-date, and that the initial processing cost is lower. Additionally, data that has partially bad quality can be handled easily as long as the problems do not affect the queries at hand. The main drawback is that queries will be much slower. To avoid this, a mixed implementation can be the solution.

Acknowledgements

This work was supported by the Danish Research Council for Technology and Production, through the framework project “Intelligent Sound” (FTP No. 26-04-0092), and the Spanish National Research Project TIN2008-01825/TIN.

References

Abelló, A. (2002). *YAM2: A Multidimensional Conceptual Model*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya (Spain).

Abelló, A., Samos, J., and Saltor, F. (2006). *Yam2: a multidimensional conceptual model extending UML*. Information Systems, 31(6), pages 541-567.

Baader, F. and Sattler, U. (2003). *Description logics with aggregates and concrete domains*. Information Systems, 28(8), pages 979-1004.

Bao, J., Caragea, D., and Honavar, V. (2006). *Package-based description logics – preliminary results*. International Semantic Web Conference, pages 967-969.

Beyer, K., Chambérlin, D., Colby, L. S., Özcan, F., Pirahesh, H., and Xu, Y.(2005). *Extending XQuery for analytics*. In Proc. of the ACM SIGMOD International Conference on Management of Data, pages 503-514.

Borgida, A. and Serafini, L. (2003). *Distributed description logics: Assimilating information from peer sources*. Journal on Data Semantics, 1, pages 153-184.

Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., and Stuchenschmidt, H. (2003). *C-OWL: Contextualizing ontologies*. Second International Semantic Web Conference, pages 164-179.

Bruckner, R. M., Ling, T. M., Mangisengi, O., and Tjoa, A. M. (2001). *A framework for a multidimensional OLAP model using topic maps*. In Proc. of the 2nd International Conference on Web Information Systems Engineering, pages 109-118.

Calvanese, D., Giacomo, G. D., and Lenzerini, M. (2001). *A framework for ontology integration*. In Semantic Web Working Symposium, pages 303-316.

Choi, N., Song, I-Y, Han H. (2006). *A survey on ontology mapping*. SIGMOD Record, 35(3), pp 34-41.

Cuenca-Grau, B. and Kutz, O. (2007). *Modular ontology languages revisited.* In Proc. of the IJCAI-2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition.

Cuenca-Grau, B., Parsia, B., Sirin, E., and Kalyanpur, A. (2005). *Automatic partitioning of OWL ontologies using E-connections*. In Description Logics, volume 147 of CEUR Workshop Online Proceedings.

Daconta, M. C., Smith, K. T., and Obrst, L. J. (2003). *The Semantic Web: A guide to the future of XML, web services, and knowledge management*. John Wiley and Sons, Inc., New York.

Danger, R. and Berlanga, R. (2008). *A Semantic Web approach for ontological instances analysis*. Communications in Computer and Information Science, 22, pages 269-282.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag Heidelberg, Berlin.

Franconi, E. and Ng, G. (2000). *The i.com tool for intelligent conceptual modeling*. In Proc. of the 7th International Workshop on Knowledge Representation Meets Databases, pages 45-53.

Garwood, K., McLaughlin, T., Garwood, C., Joens, S., Morrison, N., Taylor, C. F., Carroll, K., Evans, C., Whetton, A. D., Hart, S., Stead, D., Yin, Z., Brown, A. J., Hesketh, A., Chater, K., Hansson, L., Mewissen, M., Ghazal, P., Howard, J., Lilley, K. S., Gaskell, S. J., Brass, A., Hubbard, S. J., Oliver, S. G., and Paton, N. W. (2004). *PEDRo: a database for storing, searching and disseminating experimental proteomics data*. BMC Genomics, 5(68).

Golfarelli, M., Rizzi, S., and Vrdoljak, B. (2001). *Data warehouse design from XML sources*. In Proc. of the 4th ACM International Conference on Data Warehousing and OLAP, pages 40-47.

Harris, S. and Gibbins, N. (2003). *3store: Efficient Bulk RDF Storage*. Proc. of the First International Workshop on Practical and Scalable Semantic Systems, volume 89 of CEUR Workshop Online Proceedings.

Horrocks, I. and Sattler, U. (2003) *Decidability of SHIQ with complex role inclusion axioms*. International Joint Conference on Artificial Intelligence, pages 343-348.

Hu, W., Qu, Y., Cheng, G. (2008). *Matching large ontologies: A divide-and-conquer approach.* Data and Knowledge Engineering, 67, pages 140-160.

Hurtado, C. A. and Mendelzon, A. O. (2002). *OLAP dimension constraints*. In Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 169-179.

Hurtado, C. A., Gutiérrez, C., and Mendelzon, A. O. (2005). *Capturing summarizability with integrity constraints in OLAP*. ACM Transactions on Database Systems, 30(3), pages 854-886.

Jameson, D., Garwood, K., Garwood, C., Booth, T., Alper, P., Oliver, S., and Paton, N. (2008). *Data capture in bioinformatics: requirements and experiences with Pedro*. BMC Bioinformatics, 9(183).

Jensen, M. R., Møller, T. H., and Pedersen, T. B. (2001). *Specifying OLAP cubes on XML data*. Journal of Intelligent Information Systems, 17(2/3), pages 255-280.

Jiménez-Ruiz, E., Berlanga, R., Nebot, V., and Sanz, I. (2007). *OntoPath: A Language for Retrieving Ontology Fragments*. In Robert Meersman and Zahir Tari, Eds., Proc. of On the Move to Meaningful Internet Systems, pages 897-914.

Jiménez-Ruiz, E., Cuenca-Grau, B., Sattler, U., Schneider, T., and Berlanga, R. (2008). *Safe and economic re-use of ontologies: A logic-based methodology and tool support*. In Proc. of the 5th European Semantic Web Conference, pages 185-199.

Kalfoglou, Y. and Schorlemmer, M. (2003). *Ontology mapping: the state of the art*. The Knowledge Engineering Review, 18(1), pages 1-31.

Köhler, J., Philippi, S., and Lange, M. (2003). *Semeda: ontology based semantic integration of biological databases*. Bioinformatics, 19(18), pages 2420-2427.

Lenz, H., Shoshani, A. (1997). *Summarizability in OLAP and statistical data bases*. Ninth International Conference on Scientific and Statistical Database Management, pages 132- 143.

Louie, B., Mork, P., Martin-Sanchez, F., Halevy, A., and Tarczy-Hornoch, P. (2006). *Data integration and genomic medicine*. Journal of Biomedical Informatics 10(1), pages 5-16.

Lu, J., Ma, L., Zhang, L., Brunner, J.-S., Wang, C., Pan, Y., and Yu, Y. (2007). *SOR: A practical system for ontology storage, reasoning and search*. In Proc. of the 33th International Conference on Very Large Data Bases, pages 1402-1405.

Lutz, C., Areces, C., Horrocks, I., and Sattler, U. (2005). *Nominals, and Concrete Domains*. Journal of Artificial Intelligence 23, pages 667-726.

Mangisengi, O., Huber, J., Hawel, C. and Essmayr, W. (2001). *A framework for supporting interoperability of data warehouse islands using XML*. In Proc. of the Third International Conference on Data Warehousing and Knowledge Discovery, pages 328-338.

Marian, A., Abiteboul, S., Cóbena, G., and Mignet, L. (2001). *Change-centric management of versions in an XML warehouse*. In Proc. of the 27th International Conference on Very Large Data Bases, pages 581-590.

Mena, E., Illarramendi, A., Kashyap, V., and Sheth, A. P. (2000). *Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies*. Distributed and Parallel Databases, 8(2), pages 223-271.

Nebot, V. and Berlanga, R. (2009). *Building Ontologies from Very Large Knowledge Resources*. In Proc. Of 11th International Conference on Enterprise Information Systems (submitted).

Nguyen, T. B., Abiteboul, S., Cóbena, G., and Preda, M. (2001a). *Monitoring XML data on the web*. In Proc. of the 2001 ACM SIGMOD International Conference on Management of Data, pages 437-448.

Nguyen, T. B., Tjoa, A. M. and Mangisengi, O. (2001b). *Meta Cube-X: An XML Metadata Foundation of Interoperability Search among Web Data Warehouses*. In Proc. of the Third International Workshop on Design and Management of Data Warehouses, volume 39 of CEUR Workshop Online Proceedings.

Pedersen, D., Riis, K., and Pedersen, T. B. (2002). *XML-extended OLAP querying*. In Proc. of the 14th International Conference on Scientific and Statistical Database Management, pages 195-206.

Pedersen, T. B., Jensen, C. S., and Dyreson, C. E. (1999). *Extending practical pre-aggregation in on-line analytical processing*. In Proc. of the 25th International Conference on Very Large Data Bases, pages 663-674.

Pedersen, T. B., Jensen, C. S., and Dyreson, C. E. (2001). *A foundation for capturing and querying complex multidimensional data*. Information Systems 26(5), pages 383-423.

Pérez J.M., Berlanga R., Aramburu M.J. and Pedersen, T.B (2008). *Integrating data warehouses with web data: A survey*. IEEE Transactions on Knowledge and Data Engineering, 20(7), pages 940-955.

Pérez-Rey, D., Maojo, V., García-Remesal, M., Alonso-Calvo, R., Billhardt, H., Martin-Sánchez, F., and Sousa, A. (2005). *Ontofusion: Ontology-based integration of genomic and clinical databases.* Compututers in Biology and Medicine, 36(7-8), pages 712-730

Pokorny, J. (2001). *Modelling stars using XML*. In Proc. of the Furth ACM International Conference on Data Warehousing and OLAP, pages 24-31.

Priebe, T., and Pernul, G. (2003) *Ontology-based integration of OLAP and information retrieval*. In Proc. of the 14th International Workshop on Database and Expert Systems Applications, pages 610-614.

Roldán-García, M. del M., Aldana-Montes, J.F. (2008). *DBOWL: Towards a scalable and persistent OWL reasoner*. In the Third International Conference on Internet and Web Applications and Services, pages 174-179.

Romero, O., and Abelló, A. (2007). *Automating multidimensional design from ontologies*. In Proc. of the 10th International Workshop on Data Warehousing and OLAP, pages 1-8.

Rubin, D.L., Shah, N.H. and Noy, N.F. (2007). *Biomedical ontologies: a functional perspective*. Briefings in Bionformatics 9(1), pages 75-90.

Schaerf, A. (1994). *Reasoning with individuals in concept languages*. Data Knowledge Engineering, 13(2), pages 141-176.

Schmidt-Schauss, M. and Smolka, G. (1991). *Attributive concept descriptions with complements*. Artificial Intelligence, 48(1), 1-26.

Simitsis, A., Skoutas, D., and Castellanos, M. (2008). *Natural language reporting for ETL processes.* In Proc. of the ACM 11th International Workshop on Data Warehousing and OLAP, pages 65-72.

Skoutas, D., and Simitsis, A. (2006). *Designing ETL processes using Semantic Web technologies.* In Proc. of the ACM 9th International Workshop on Data Warehousing and OLAP, pages 67-74.

Stuckenschmidt, H. and Klein, M. C. A. (2007). *Reasoning and change management in modular ontologies*. Data and Knowledge Engineering, 63(2), pages 200-223.

Wang, L., Zhang, A., and Ramanathan, M. (2005). *Biostar models of clinical and genomic data for biomedical data warehouse design*. International Journal of Bioinformatics Research and Applications, 1(1), pages 63-80.

Whoweda. The Web Warehousing and Mining Group. (1997). Retrieved October 14, 2006 from http://www.cais.ntu.edu.sg:8000/˜whoweda.

Xyleme (2001). *A dynamic warehouse for XML data of the Web*. IEEE Data Engineering Bulleting, 24(2), pages 40-47.