Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100412

A Conceptual Data Model for the Automatic Generation of Data Views

Félix Fernández-Peña[1,*], *Pilar Urrutia-Urrutia*[1], *René Cañete*[2], *Rolando Acosta-Sánchez*[3], *Cornelio Yañez-Márquez*[4] *and Jyrki Nummenmaa*[5]

[1] Universidad Técnica de Ambato, Ecuador
[2] Universidad Tecnológica Israel, Ecuador
[3] Instituto Superior Politécnico José Antonio Echeverría, Cuba
[4] Instituto Politécnico Nacional, Mexico
[5] University of Tampere, Finland

Received: 25 Mar. 2016, Revised: 11 May 2016, Accepted: 12 May 2016
Published online: 1 Jul. 2016

Abstract: Traditionally, data views are seen as static, syntactically correct, data sets. The semantics of the data is not explicitly encoded in Relational Databases (RDB) but implicitly on the applications. However, the needs to create context-aware browsing methods in changing scenarios are demanding more flexible mechanisms. Data views need to be semantically enriched for capturing the real world changes. In this paper, we evaluate the definition of a conceptual data model for the automatic ontology-based building of data views. Our tests show that our approach increases the flexibility of information systems while decreases their maintenance costs.

Keywords: Conceptual data model, description logic, relational data, semantic data views

1 Introduction

The corporate environment is increasingly dependent on the speed and accuracy of information retrieval [1]. Reliable and up-to-date data, along with the necessary knowledge needed to interpret it, are probably the most important resources in this scenario [1,2,3].

Operational data retrieval has a key role in corporate information systems as they act as an interface to data and information. In this context, classical database design strategies, based on the a priori definition of data views to be finally integrated in the design of the global databases, are not appropriate for the design needs of modern, highly dynamic information systems [2,4].

An efficient system for retrieving information must be flexible, being able to adapt to changes in the information system life cycle [2,3,4]. New operational data, new data analysis and changing the understanding of data are common, and naturally accepted, as well as the fallibility of knowledge is [1]. However, adapting information systems to changing environments generally requires a lot of manual effort [2,3].

The recoding of traditional information systems is required when the interpretation of data changes [3]. In this context, Chen et al. demonstrated the usability of flexible, multiple views-based system to support awareness for cooperative design [4]. Nevertheless, they do not tackle the problem that raises in complex scenarios because of the intractability of the implicit semantics.

By making the semantics of RDB explicit, we propose to reduce maintenance costs and to increase the flexibility of data retrieving in information systems. Since Resource Description Language (RDF) makes it possible to define the meaning of data in a machine readable form [5], it seems that the semantic web technologies could be helpful in the alignment of RDB towards the semantic dimension of data views manageability.

A lot of attention has been paid to the semantic enrichment of data stored in RDB [6,7]. The evolution of RDF into Web Ontology Language (OWL) allows a richer semantic description based on Description Logic [8]. OWL is a formal language for representing ontologies in the Semantic Web [8]. This language has been used in many specific scenarios for the construction of flexible data semantic models [9,10,11].

Interest in mapping relational data to RDF is increasing for the purpose of publishing linked data [12].

* Corresponding author e-mail: fo.fernandez@uta.edu.ec

In this direction, SPARQL is the World Wide Web Consortium (W3C) recommended query language for information retrieving from RDF documents. In the semantic web vision, SPARQL is considered the theoretical equivalent to SQL in relational databases [11]. However, as Hert et al. did in 2010 [13], we consider that converting relational data to RDF is often not feasible.

The semantics of data is not explicit in RDB. Then, we propose not to transform databases into semantically aware data but to define an ontology-driven descriptive programming method for data retrieving. In this paper, we formally define the semantic structure of generic data views. Our design is based on the description of the relationships of the data views V with the set of data columns D_i, such that our aim is to define a conceptual data model not in the syntactic but in the semantic dimension of data.

Thus, when updating the understanding of the domain of discourse there is no need to recode the data retrieving procedures but to transform the semantic representation of retrievable data. As a consequence, the automatic generation of data views is achievable and data are actually charted taking into account the relevance and the actual meaning of data.

We have constructed a functional prototype of the data views generator for evaluating capabilities of data presentation and browsing when using our conceptual data model. Our comparison with a traditional implementation of Relational Database Management Systems (RDBMS) shows that our approach could be beneficial for data retrieving in information systems when an accurate semantic description of the RDB exists.

The paper is organized as follows. First, the related work is briefly reviewed. In section 3, we explain how our method works and in the next section we formally define our conceptual data model for describing the semantics of data views. In section 5, we analyze usability issues of information gathering in the implementation of a prototype. A scenario of proof and validation of the proposal is described, as well. The discussion of results focuses on data retrieving using our approach and the traditional SQL-based report generation using RDBMS tools. Finally, conclusions and future work are analyzed in section 6.

2 Related Work

2.1 View-Based Semantic Browsing

Defining and using views in information seeking tasks has been the focus of several researchers. Dichev and Dicheva proposed a view-based semantic search and browsing model [14]. Their proposal is based on the semantic description of views derived from user tasks or goals. The customization of the user interface for retrieving data improves since information about the user profile is made explicit in context. However, the implicit relevance of the kind of data stored in the RDB keeps hidden to the software. The importance of the context-awareness in the application layer of view-based initiatives is mentioned by Hong et al. in their survey of context-aware systems [15] and, more recently, by Namiot et al. [16]. Later on, Bolchini et al. have proposed CARVE, a methodology for context-aware view definition [2]. CARVE is based on a context model, on guidelines to define partial views and on a set of operators for view composition in context. This methodology means an step forward in flexible data views generation but the linkage between context-aware data view definition and the view-based semantic formal definition is not well addressed in the literature until now.

2.2 Ontology-Based RDB Schemas

Transforming the vast amount of data, currently residing in Relational Databases, into semantically aware data structures has been previously identified as a necessity [6, 7, 8, 9, 17, 18, 19]. Ontology-based RDB schemas have been proposed for specific scenarios [6] and distributed environments [7, 8, 20].

Barsalou et al. have worked in a semantic data model for enhancing relational databases. Their proposal was based in structuring and manipulation tools that take more domain information into account and provide the user with an appropriate level of abstraction in the studied scenario [21]. Nevertheless, the applicability of their tools was restricted to the area of immunogenetics. Almost twenty years later, Sun and Fan recalled that the semantic extraction is essential for the semantic interoperability in multi-enterprise business collaboration and they proposed a method for acquiring semantics from heterogeneous data schemas [7]. Sun and Fan proposed a unified syntax-independent conceptual model that showed to be extensible and flexible in multi-enterprise business collaboration environments [7]. Further, Guido and Paiano proposed to take the integration of information systems as a whole to a semantic dimension [22].

More recently, Song et al. proposed SIL (a Semantic Information Layer) as mediation media among heterogeneous database systems [20]. A dynamic multi-strategies ontology alignment with automatic matcher selection and dynamic similarity aggregation allowed them mapping data sources for retrieving distributed data. All these have been important contributions in data retrieving. However, these studies are not focused on the semantic enrichment of RDB data views but on overcoming the gap of conceptual heterogeneity. To our knowledge, the semantic web technologies have not been widely used in the generation of data views and it seems there is no consensus yet on how to fully take RDB into the semantic dimension of data management.

2.3 RDB Semantic Mapping

Many mapping languages and approaches were explored leading to the ongoing standardization effort of the W3C carried out in the RDB2RDF Working Group [17]. RDOTE is a sound proposal for the automatic and custom mapping and transportation of data residing in RDB into RDF [19].

Agus et al. incorporated *concept hierarchy* as background knowledge for the OWL ontologies extraction on top of RDB [9]. The RDB2OWL language [10] reuses the OWL ontology structure as a backbone for mapping specification by placing the database link information into the annotations of ontology classes and properties.

Hert et al. have worked in updating data stored in RDB from the semantic dimension of data management. They emphasized in the importance of formalizing a semantics-based RDB management towards flexibility [13]. Wu et al. have been working in semantic query, search and navigation services by dynamically mapping SPARQL queries to SQL queries. Their proposal considers using a concepts-ranking mechanism to provide more accurate and reliable search results for the users [18]. More recently, Aufaure et al. [23] propose using continuous queries, data summarization and matching for working with data semantics in data retrieving processes. The authors emphasize the importance of context and techniques they propose to integrate have been broadly used in semantic web initiatives [15,22,20]. Nevertheless, the lack of an experimental evaluation of results makes arguable the business intelligence inspiration of their proposal.

In general, we consider that 1) RDB mapping into the semantic web should not be syntactically limited to one-to-one relations between database tables and ontology concepts and that 2) the implicit semantics enrichment of RDB should be taken into account and made explicit. Further, the translation of SPARQL queries into SQL is far from trivial when the use of grouping or mathematic functions is needed to calculate actual values for data fields defined in the logical design of RDB [11].

This limitation could be related to the fact that mathematic functions and aggregates have not been actually included in the SPARQL language until a few months ago [24]. Anyway, two independent studies have shown that RDB to RDF mapping systems executing SPARQL on relational databases are several orders of magnitude slower than executing the semantically equivalent SQL query directly on the RDBMS [25].

Instead of migrating available legacy data in relational database into ontologies, it seems an option to keep using SQL for what it has proved to be good at and to explicitly link concepts from the semantic dimension of data manageability to relational databases.

2.4 Description Logic

Description Logic (DL) is defined as a family of formalisms for knowledge representation. Relevant concepts, for an specific knowledge domain, are defined and then used in declaring specific properties of objects and individuals [26].

The basic types of a *concept language* are *concepts* and *roles*. A *concept* is a description gathering the common properties among a collection of individuals; from a logical point of view, it is a unary predicate. Inter-relationships between these individuals are represented by means of roles (which are interpreted as binary relations).

A knowledge base, in this context, is a finite set $\sum$ of terminological axioms (often called T-Box) and a set of assertional axioms or assertions (often called A-Box). A DL system is characterized by four aspects [26]:

1. The set of constructs constituting the language used for building the *concept expressions* and *role expressions*.
2. The kind of assertions allowed in the T-Box (assertions on concepts).
3. The kind of assertions allowed in the A-Box (assertions on individuals).
4. The inference mechanisms provided for reasoning on the knowledge bases expressible in the system.

OWL was designed such that it is able to encode database schemas expressed in the most interesting Semantic Data Models and Object-Oriented Data Models [8,20]. OWL was used as the basic representation language of the terminology we defined.

3 Outline of Method

The motivation behind our work is to support data retrieving procedures in terms of the semantics (meaning) of data views. Our work is inspired on the design of ontologies for OLAP data integration, previously developed by one of the authors (Nummenmaa) with Niemi et al. [3]. In previous work, we published partial results in the definition of ViewOnto, an ontology for the semantic description of data views [27][28]. In this paper, we now formalize the conceptual data model behind the design of ViewOnto and we analyze the results of the assessment of the model usability through experimentation.

The framework which supports the implementation of the model we propose is structured in three layers with different levels of abstraction. The top layer is defined as a terminology $\sum$ which is used for describing data views of relational databases. This terminology is stored as the T-Box of the OWL-encoded ontology that we call *ViewOnto*. The middle layer is structured as one A-Box β_i for each RDB which is used in the information recovery processes. β_i instantiates the formal semantic schema in an specific context i. Lastly, instances of data items (the actual data) are retrieved from RDB in the bottom layer Y. A set of web services in Y interpret the mapping of the semantics, formally declared in $\sum$, to the actual data of a specific β_i These web services provide data to an automatically generated user interface for browsing data and for generating and exporting data reports. Figure 1 illustrates how these three layers interact among each other in the case of World trade data (RDB *Trade*).

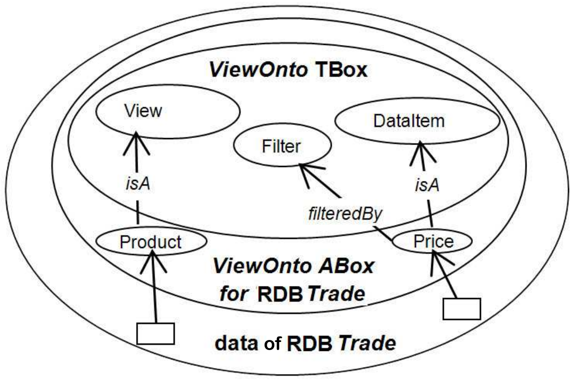


Fig. 1: Interaction among layers of the framework in the case of the RDB *Trade*

World trade data is used to illustrate our method. This data contain auto-generated import/export figures. Figure 2 shows the Syntactic RDB Schema. The example aims to demonstrate the different relationships among the semantic and syntactic layers of language formalized with *ViewOnto*. The implementation of a prototype for the defined ontology-based procedure to retrieve information was tested using this data. Results are analyzed in section 5.

In our example, the data items stored in the RDB characterize products, importers, exporters and trades among them. As a whole, our data gathering procedure works as follows:

1. Row data is available in the RDB through specific SQL queries.
2. A conceptual definition of the data views is made explicit by defining the corresponding asserting axioms in a specific A-Box of *ViewOnto*.
3. A generic application renders the appropriate user interface for browsing, filtering and charting information. Rendering process takes into account the semantics of data views and its mapping into syntactically correct SQL sentences.

The interaction among the layers of the framework replicates seamlessly when working with more than one database. Our prototype is a generic implementation of the user interface for data retrieving. Its implementation is described in section 5.

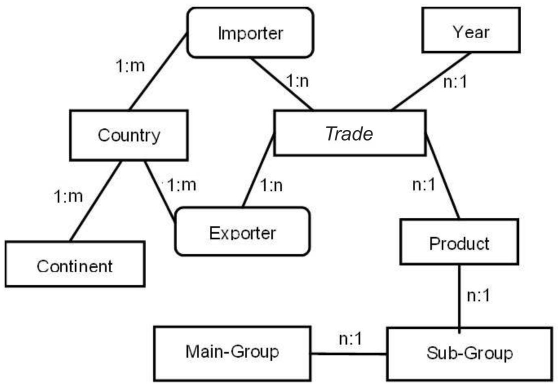


Fig. 2: RDB Model of world *trade* data

4 Modeling the Semantics of Data Views

In this section, we introduce a conceptual model for representing the semantics of data views. A formal schema is used to describe abstract properties of data views, their data items, and semantically enriched relationships among data items and relationships among data views.

4.1 Basic Modeling Language

Let us first introduce the notational conventions used in the definitions. Letters A, B are used as *atomic concepts* and C, D for *concept descriptions*. For *roles*, we use the letters R, S. In concrete examples of our study case, *concept names* start with an uppercase letter followed by lowercase letters (e.g., *View*, *DataItem*), role names start with a lowercase letter (*e.g.*, *relatedTo*), and individual names are all uppercase (*e.g.*, *TRADE*, *EXPORTER*).

An interpretation I consists of a non-empty set Δ^I (*the domain of the interpretation*) and an interpretation function, which assigns to every *atomic concept* A a set $A^I \subseteq \Delta^I$ and to every atomic role R a binary relation $R^I \subseteq \Delta^I \times \Delta^I$ [26]. In the case of *datatype properties* (roles that take values in the domain of datatypes like *integer*, *string*) $R^I \subseteq \Delta^I \times \Delta^I_D$ [26], where D represents a specific *datatype*.

Syntax and semantics of these axioms can be found in Table 1. An interpretation I is called a model of an axiom if it satisfies the statement in the last column of the table.

An equality whose left-hand side is an *atomic concept* (*role*) is called *concept* (*role*) *definition*. A set of axioms of the form $R \sqsubseteq S$ where both R and S are atomic is called *role hierarchy*. Such a hierarchy obviously imposes restrictions on the interpretation of roles. Axioms of the form $C \sqsubseteq D$ for a complex description C are often called *general inclusion axioms*.

Table 1: Terminological and assertional axioms

Name	Syntax	Semantics
Concept inclusion	C⊑D	$C^I \sqsubseteq D^I$
Role inclusion	R⊑S	$R^I \sqsubseteq S^I$
Concept equality	C≡D	$C^I = D^I$
Role equality	R≡S	$R^I = S^I$
Concept assertion	$C(a)$	$a^I \in C^I$
Role assertion	$R(a,b)$	$(a^I, b^I) \in R^I$

With respect to the formal apparatus, we strictly follow the concept language formalism accepted by the DL community [26]. A list of the DL concept constructors used in our approach is shown in Table 2.

Table 2: DL Concept constructors

Name (syntax)	Semantics
Top ($\top$)	Δ^I
Bottom ($\perp$)	$\varnothing$
Intersection ($C \sqcap D$)	$C^I \cap D^I$
Union ($C \sqcup D$)	$C^I \cup D^I$
Value restriction ($\forall R.C$)	$\{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$
Existential quantification ($\exists R.C$)	$\{a \in \Delta^I \mid \exists b.(a,b) \in R^I \wedge b \in C^I\}$
Exact number restriction ($= nR$)	$\{a \in \Delta^I \mid \lvert\{b \in \Delta^I \mid (a,b) \in R^I\}\rvert = n\}$

4.2 Encoding the Semantics of RDB Data Views

It is shown how the T-Box of *ViewOnto* can be expressed in the formal logic of OWL. *ViewOnto* provides a terminology to be used by a general data retrieving method.

In the following, we formally define the translation of the description of RDB data views to the formal logic of an OWL knowledge base.

Definition 1: Translation.

Let V be the data views of an information system and let D be the set of data items in those data views. Translation is defined as the mapping of V and D into a terminology $\sum$ for describing V, D, and the relationships $V \times D$ and $D \times D$.

The axioms defining $\sum$ are grouped in four categories:

A1. Concept Hierarchy:

$$V, D, I, S, R, F \sqsubseteq \top \tag{1}$$

View V, *Identifier I*, *Descriptor D*, *DataItem S*, *Relationship R* and *Filter F* are the basic concepts of the terminology $\sum$.

$$R_v, R_s \sqsubseteq R \tag{2}$$

Both, the relationships among views (R_v) and the relationships among data items (R_s) are sub-concepts of the generic relationship (R).

$$S_d, S_n, S_i, S_b, S_s \sqsubseteq S \tag{3}$$

Semantic datatypes include the concepts *DateDataItem* S_d, *NumberDataItem* S_n, *ImageDataItem* S_i, *BooleanDataItem* S_b and *StringDataItem* S_s. All these data items are sub-concepts of data item (S). The appropriate *DataItem* is decidable by the corresponding field type in the syntactic layer. By instantiating these concepts, a more accurate behavior is automatically achievable in the user interface of the data view generator (see section 5).

$$F_i^d \sqsubseteq, F : 1 \le d \le m, 1 \le i \le n \tag{4}$$

There are n different valid *filters* for each of the m supported datatypes of the filtered *data items*. For instance, *lessThan*, *greaterThan*, *equalTo* are valid *filters* for numeric type *data items*. Meanwhile, *startsWith*, contains, endsWith are valid *filters* for string type data items.

A2. Concept Equality:

$$V \sqcap S \sqcap R \sqcap D \sqcap I \sqcap F \equiv \perp \tag{5}$$

Concepts *V, S, R, D, I, F* are disjoint sets.

$$S_d \sqcap S_n \sqcap S_i \sqcap S_b \sqcap S_s \equiv \perp \tag{6}$$

Concepts S_d, S_n, S_i, S_b, S_s are disjoint sets.

A3. Concept Definition:

$$\begin{aligned} V \equiv &= 1title \sqcap \\ &= 1mappedTo \sqcap \forall isIdentifiedBy.I \sqcap \\ &= 1isIdentifiedBy \sqcap \forall isDescribedBy.D \sqcap \\ &\forall isRelatedBy.R_v \end{aligned} \tag{7}$$

A View V is defined as the concept with one (and only one) *title*, *mappedTo* and *isIdentifiedBy* roles. V is identified by an *Identifier* I. Further, V may be related (or not) with one or more *Descriptors* D and *Relationships* between *Views* R_v by instantiating the *isDescribedBy* and *isRelatedBy* roles, respectively.

$$\begin{aligned} I \equiv &= 1title \sqcap \forall isIdentifiedBy.S \sqcap \\ &= 1isIdentifiedBy \sqcap \\ &= 1isAlternative \sqcap \forall isRelatedBy.R_D \end{aligned} \tag{8}$$

A semantic *Identifier* is defined as the concept I with one (and only one) instance of the roles *title*, *identifiedBy* and *isAlternative*. Further, a concept I may instantiate the role *isRelatedBy* (or not) in the range of instances of R_D. All instances of *identifiedBy* and *isRelatedBy* roles that apply to I take values in the range of the set of instances of the concepts S and R_D, respectively.

$$\begin{aligned} D \equiv &= 1title \sqcap \forall describedBy.S \sqcap \\ &= 1describedBy.S \sqcap \\ &= 1isAlternative \sqcap \forall isRelatedBy.R_D \end{aligned} \tag{9}$$

A *Descriptor* is defined as the concept D, which instantiates once (and only once) the roles *title*, *describedBy* and *isAlternative*. The roles *describedBy* and *isRelatedBy* only take value in the range of S and R_D instances, respectively. All these restrictions are mandatory in a *Descriptor*.

$$\begin{aligned} S \equiv &= 1title \sqcap \\ &= 1fieldName \sqcap \forall isFilteredBy.F \end{aligned} \tag{10}$$

A *DataItem*, defined as the concept S, instantiates once (and only once) the roles *title* and *fieldname*. Further, S may instantiate (or not) the role *isFilteredBy*. If instantiated, one or more declarations of *isFilteredBy* apply to S but it is mandatory for all of them to be in the range of the instances of the concept *Filter* (defined below in axiom 14).

$$R \equiv = 1title \sqcap = 1joinCond \tag{11}$$

A *Relationship* is a concept R, characterized by one instance (and only one) of the roles *title* and *joinCond*. As a generic relationship in $\sum$, R establishes a semantic liason among other concepts.

$$R_v \sqsubseteq \forall relatedTo.V = 1relatedTo \tag{12}$$

The concept *InterViewRel* *(R_v)* refers to a relationship among two different data views V_i and $V_j : i \neq j$. Every R_v is a relationship R (axiom 2). The role *relatedTo* links the *relationship* R_v (associated to V_i by the role *isRelatedBy*) to a different *View* V_j. R_v may instantiate the role *relatedTo* (or not) as many times as necessary. Meanwhile, there is only one instance of the role *relatedTo* in the definition of an instance of R_v.

$$R_D \sqsubseteq = 1relatedTo \sqcap \forall relatedTo.(I \sqcup D) \tag{13}$$

The concept *InterDataRel* (R_D) is defined as the relationship (see axiom 2) of one *Identifier* or *Descriptor* A with a different *Identifier* or *Descriptor* B. One (and only one) instance of the role *relatedTo* defines R_D and this role takes value in the range of the set of the concepts *Identifier* or *Descriptor*. The concept A references R_D by means of the role *isRelatedBy* (see axiom 8).

$$F = 1title \sqcap = 1sqlCond \sqcap = 1clientFiltering \tag{14}$$

A *Filter* is a concept F with one (and only one) instance of the roles *title*, *sqlCond* and *clientFiltering*. F is defined for setting which are valid filters for data items.

The declaration of the data type determines a set of filters applicable for the *data item*. Thus, the declaration of valid filters in the definition of a *DataItem* adds a more restrictive condition for filtering over the match between the semantic and syntactic level of language.

A4. General Inclusion:

Corresponding roles are properly restricted with proper datatypes.

$$T \sqsubseteq \forall title.string \tag{15}$$

The role *title* identifies concepts with an *string* expression in natural language.

$$T \sqsubseteq \forall mappedTo.string \tag{16}$$
$$T \sqsubseteq \forall isAlternative.boolean \tag{17}$$
$$T \sqsubseteq \forall fieldName.string \tag{18}$$
$$T \sqsubseteq \forall joinCond.string \tag{19}$$
$$T \sqsubseteq \forall sqlCond.string \tag{20}$$

The role *sqlCond* sets the sentence in the SQL syntax corresponding to the “WHERE” condition used in the information recovery processes.

$$T \sqsubseteq \forall clientFiltering.boolean \tag{21}$$

When to filter in the client side or the server side of the application is decided by setting the value of the role *clientFiltering*.

The implications of the defined terminology for the automatic generation of data views are analyzed in section 5.

4.3 Describing Views of the RDB Trade

We have given a general/abstract formalization of our conceptual data model. The semantic terminology of the data views of a particular RDB is created by instantiating *ViewOnto* in an specific scenario. This is shown in this section through the formal description of the A-Box for the *Trade* study case, partially shown below:

Concept Assertions:

V(TRADE).
V(EXPORTER).
I(TRADEID).
D(TRADEVALUE).
D(TRADECOUNT).
D(IMPORTERNAME).
D(EXPORTERNAME).
S_n(TRADEID_S).
S_n(TRADEVALUE_S).
S_n(TRADECOUNT_S).
S_s(IMPORTER_S).
S_s(EXPORTER_S).
R_v(T_E).
R_D(TV_TC).

Role Assertions:

title(TRADE,“OFFICIAL TRADE”).
isIdentifiedBy(TRADE,TRADEID).
identifiedBy(TRADEID,TRADEID_S).
describedBy(IMPORTERNAME, IMPORTER_S).
describedBy(EXPORTERNAME, EXPORTER_S).
describedBy(TRADEVALUE, TRADEVALUE_S).
describedBy(TRADECOUNT, TRADECOUNT_S).
title(TRADEID_S, “Identifier”).
title(IMPORTER_S,“Imported by”).
title(TRADEID_S, “Id”).
title(TRADEVALUED_S,“Amount”).
isDescribedBy(TRADE, IMPORTERNAME).
isDescribedBy(TRADE, EXPORTERNAME).
isDescribedBy(TRADE,TRADEVALUED).
isDescribedBy(TRADE,TRADECOUNTD).
isRelatedBy(TRADE,T_E).
isRelatedBy(TRADEVALUED, TV_TC).
relatedTo(T_E,EXPORTER).
relatedTo(TV_TC, TRADECOUNT).
isAlternative(TRADEVALUE,false).
isAlternative(TRADECOUNT,true).

By classification of these concept (*role*) assertions of the A-Box into the T-Box, an automatic inference process is able to determine that:

1. The *View* TRADE is identified by the *Identifier* TRADEID.
2. TRADE is described by the *Descriptors* IMPORTERNAME, EXPORTERNAME, TRADECOUNT and TRADEVALUE.
3. TRADEVALUE is considered fundamental data in the interpretation of TRADE as a *View* but TRADECOUNT is not (because of the true/false value of the role isAlternative).
4. When analyzing the *View* TRADE, the *View* EXPORTER pops up as containing potentially relevant information. In the same way, the correlation among figures of the *Descriptor* TRADEVALUE and the *Descriptor* TRADECOUNT is understood as significant for decision making.

Each *View* becomes into a unique instance in *ViewOnto*. However, this instance is determined (in the syntactic layer) by combining one or more field names, grouping functions and calculated fields from *n* tables in SQL syntax. I.e., *tradeValueD* and *tradeCount* might be calculated by using the SQL functions *sum*() and *count*(), respectively (to be specified in the role *mappedTo* of the *View* TRADE).

Actual data of a *view V* is seen in a tabular data report. Mathematically, it means the matrix derived from a set of *n* rows of data (filtered or not) in one or more of the *m* semantically relevant data items identifying and describing *V*.

In general, one *View* may be related with one or more tables, and viceversa. I.e., the name of exporters is considered a *Descriptor* in the *View* TRADE (more than one table are related to the *View*). Likewise, the table *exporter* is related to the *View* EXPORTER as well. The definition of the A-Box of *ViewOnto* for an specific scenario is easily updated by making changes to the corresponding OWL file.

No prescriptive programming but the update of the corresponding instance of *ViewOnto* is required in order to adapt *ViewOnto*-based data retrieval tools to changing scenarios and to keep using these tools with different RDBs. The scope of possible changes includes: *new operational data*, *new data analysis* and *changing the understanding of data*. Below, the scope of the possible changes is discussed, giving examples in the context of the study case.

New operational data: The domain of discourse is extended by defining new *views* and/or new *Descriptors* of existing *views*. The process involves 1) the definition of new *Descriptors*, *data items*, the corresponding semantic *relationships* and *filters* and 2) the re-definition of the role *mappedTo* of involved *views* (the liaison of the semantic description of data with the SQL syntax). Alternatively, the role *mappedTo* may point out to a web service that encapsulates the syntactic access to the actual data. This way, any data access optimizing tools -like making use of stored procedures- may be transparently used.

For instance, the *view* TRADE in our study case retrieves information about trades per exporters. For doing so, the role *mappingTo* utilizes a grouping function and it defines a calculated field for totalizing the amount of money that any exporter receives. If including the number of trade operations is considered useful for data analysis, the user needs to update the corresponding role *mappedTo* of the *view* TRADE and to define the corresponding *Descriptor*, *data item* and the applicable *filters* and *Relationships* of the new *Descriptor* with other *Descriptors* in the scenario. Alternatively, by having the accurate definition of the necessary *Descriptors*, hiding/showing any of them only requires to set the role *isAlternative* to the proper value.

New data analysis: If the reinterpretation of a specific *View* is possible from a different point of view, it makes sense to assess the definition of each of the possible interpretations as a different *View*. The definition of any of the *Views* in this case requires similar effort. The steps to follow are like those described in the case of *new operational data*.

Further, custom *filters* may be defined taking into account the nature of the operational data. For instance, a *custom filter* could be defined for filtering exporters per region (by using the data of the continent that they belong to). This *filter* could be used for classification. By defining this *custom filter*, clusters of exporters’ data may be generated as new, useful *views*.

Changing the understanding of data: Depending on the interpretation of data, it could be necessary to create new *Views*, *Descriptors*, *Filters*, *Relationhips* between *Views* and *Relationhips* between *Descriptors*. In a different scenario, some of the definitions may be deprecated as well. Further, the labels that identify them may change. For instance, every value that the role *title* takes in the instance of *ViewOnto* could be translated into another language. This way, the usability of the retrieved data for users that speak the alternative language will be substantially increased.

5 Model's Experimental Assessment

A prototype was built in order to evaluate the feasibility of the proposed model. The implementation basically follows these steps:

1. An A-Box matching *ViewOnto*'s T-Box is loaded into the prototype.
2. The value of the role *title* for every *View* instance in the A-Box is rendered as a hotspot in the interface. Every hotspot becomes into an access point to information.
3. The role *mappedTo* is used by the data recovery processes in order to request actual data from the RDB. The user interface is rearranged and the retrieved data are shown.
4. The user makes use of related views, related data items and defined filters for expanding the search spectrum.

The users of the application start retrieving data after loading a copy of an A-Box, preferably created by an Information Engineer with a sound knowledge on the design of the operational database been used. This A-Box is what we call: *main A-Box*. Later on, users customize the semantic description of the *views* based on their own experience and preferences retrieving data. Changes are saved in what we call a *custom A-Box*. Contradictions in the interpretation of data may turn up. As a shared and computationally treatable language is used, a simple inference process might assist in detecting such contradictions. Off-line inferencing for supporting the update process of *ViewOnto* seems to be a natural line of future work.

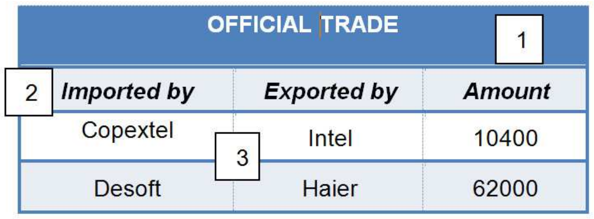
In a first experiment, 35 PhD students were asked to propose the *main A-Box* to be used for retrieving data from the RDB *Trade*. We corroborated that there were not two *A-Box* proposals that were actually the same among those elaborated by the students. Nevertheless, 33 *A-Box* proposals were actually correct, as they required no changes in order to be used for retrieving data views. Due to this result, it seems that different accurate interpretations of reality may actually coexist while working with a common scenario.

In a second phase of the experiment, every student was assigned with one *A-Box* of those that were previously proposed as *main A-Box*. None of them received the ontology proposed by him/her-self. Then, these students were requested to evaluate the data retrieving capabilities of the prototype.

69% of the students considered that the data views automatically generated after loading the corresponding *A-Box* were covering the real needs of potential users. It means that even when the ontology they used was not of their own design, they were able to successfully retrieve pertinent data views.

95% of the students proposed minor changes to the data views (mainly changes in the set of values of the role *title*), corroborating that it is common to find different interpretations of the reality. Meanwhile, 31% of the students proposed major changes (i.e., deleting/creating new data views). All major changes were feasible by transforming the ontology into a *custom A-Box*. The successful adaptability of *ViewOnto* was considered as partial validation of its flexibility and facility of maintenance.

Next, we discuss the targeted application domain. Figure 3 illustrates the matching of data into the *view* TRADE for the study case. The user interface of the implemented prototype combines three aspects, as depicted by the squared hints: 1) "Official Trade", the role *title* of the *View* TRADE, heads the user interface, 2) the value of the role *title* of each *Descriptor* of the *View* TRADE (with the role *isAlternative* in *false*) heads each column of the tabular data report, and 3) each row contains the actual data comming from the RDB *Trade*.



OFFICIAL TRADE

Imported by	*Exported by*	*Amount*
Copextel	Intel	10400
Desoft	Haier	62000

Fig. 3: *View* TRADE in the user interface of the prototype

Data retrieving and browsing is context-aware. When a user hovers over a data item value and activates the contextual menu two different options appear in the user view, as shown in Figure 4. The user then may ask for related data and the user may filter the data in the tabular data report.

Figure 4 illustrates the steps for browsing data (from TRADE to EXPORTER and then to PRODUCT) while the user goes asking for related data. Data are charted in terms of fundamental *Descriptors* (with the role *isAlternative* in "*false*") in the interpretation of the related *Views*. It is important to note that not only related *Views*, but related *Descriptors*, are available when activating the *related to* contextual submenu. *Descriptors* not shown by default in the tabular data report are marked with the role *isAlternative* in "*true*". This is the case of the *Descriptor* labeled "Trade Count" (because of the value of its role *title*), related to the *Descriptor TradeValueD* of the *View* TRADE.

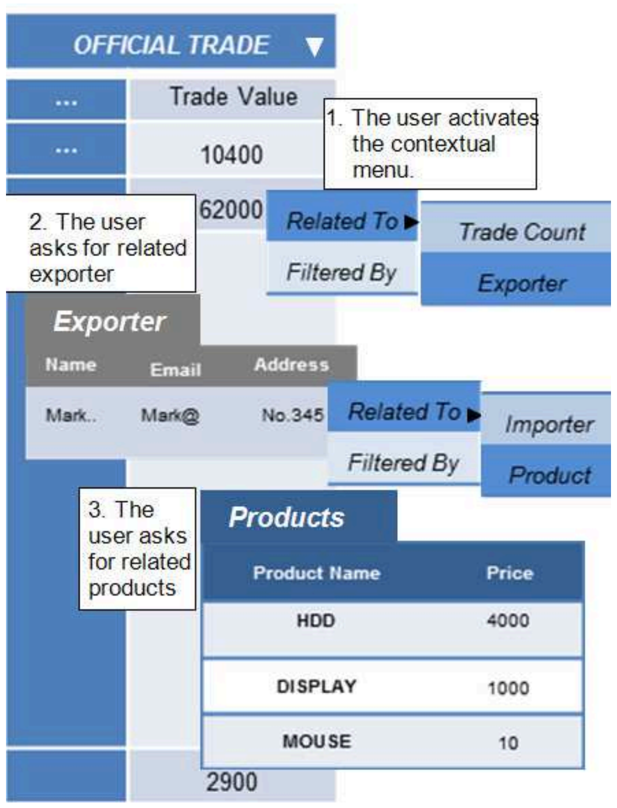


Fig. 4: Semantic navigation in data retrieving

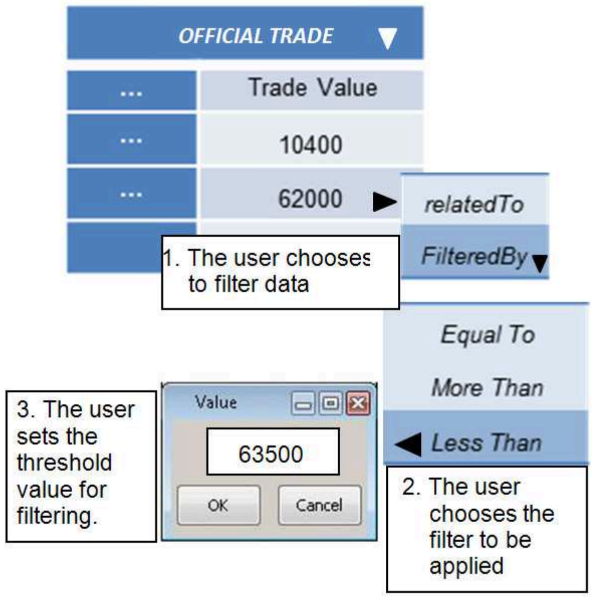

OFFICIAL TRADE
...
Trade Value
10400
62000
relatedTo
FilteredBy
1. The user chooses to filter data
Equal To
More Than
Less Than
Value
63500
OK
Cancel
3. The user sets the threshold value for filtering.
2. The user chooses the filter to be applied

Fig. 5: Applying filters to data

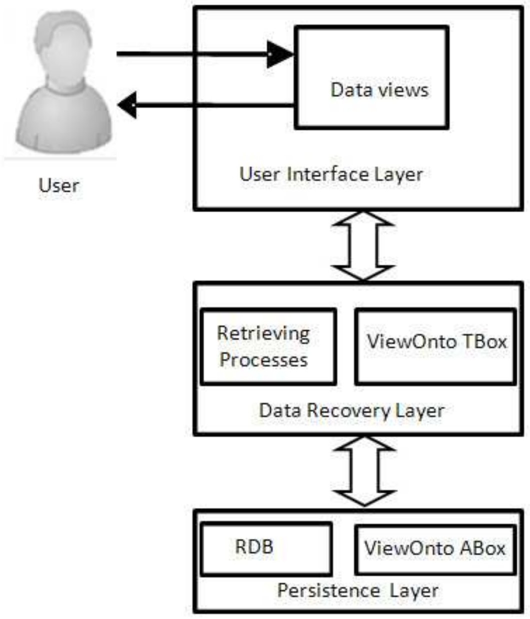


Fig. 6: Prototype software architecture

Figure 5 shows the steps for filtering data in the tabular data report of the user view while using the A-Box for the RDB *Trade*. The options of the contextual menu are automatically generated based on the defined *isFilteredBy* relationships with classes *Filter* defined in *ViewOnto*.

Querying an RDF document is considerably slower than using SQL statements to gather information from a RDB, as proved on experimentation [25]. In our case, *ViewOnto* is not populated but used as semantic schema for deciding which SQL statement to execute. This way, the efficiency of data retrieving is not affected.

The data retrieving process is located as close as possible to the data source. Then, data items are transmitted through the network to the client's user interface, where they are properly displayed. An overview of the software architecture of the prototype implementation is shown in figure 6.

A second experiment was designed in order to compare the data retrieving process using our proposal against using a traditional RDBMS. A group of 256 undergraduate students of Software Engineering participated in the evaluation. All chosen students have experience in database design and RDBMS implementation in Java. A computer with an i3 processor, 2 GB RAM and enough hard drive space was assigned to every student. Each student received the code of a RDBMS implemented in Java and our *ViewOnto*-based data retrieving prototype, both of them working with the

RDB *Trade*. For changes in the Java application, they used the IDE Eclipse.

Each of the students received the same assignment of ten data retrieving operations to be implemented, with different complexity in the required SQL syntax and different complexity in the semantics of data. The percentage of operations requiring changes in the encoding of the semantics is related with the expressive power of *ViewOnto*. Minor changes are related to label changes and setting on pre-defined filters and/or relationships among concepts. Major changes involve the extension of concepts into new ones for which SQL expressions are required in order to retrieve actual data from the database. Since a well designed *A-Box* is expected to be defined as *main A-Box*, it is presumable that less than a half of different retrieving services will require actual changes in the ontology. As a matter of fact, in the previous experiment updates to the ontology were proposed only in 31% of all cases. The list of tasks was reviewed by three information retrieval specialists in order to validate their relevance in context. Next, there is the list of tasks involved in the experiment:

1. Include the trade value of sales per pair of exporter/importer in the *View* “OFFICIAL TRADE”.
2. Delete the column “count of trades” per pair of exporter/importer in the *View* “OFFICIAL TRADE”.
3. Filter data retrieved in the *View* “OFFICIAL TRADE” for an specific exporter.
4. Retrieve detailed data of the previously specified exporter.
5. Retrieve the products exported by this exporter.
6. Filter the list of trades for total amount of sales greater than a specified value.
7. Define an alternative *View* “Trade Stats per Year” for counting trades of each importer per year.
8. Define “Trades with” as the relationship of every exporter with those importers to whom that importer has had a trading relationship.
9. Allow filtering the *View* resulting on activating the *Relationship* “Trades with” for those exporters from a specific country.
10. Filter the data view “Trade Stats per Year” for a specific year or later.

All students were instructed in how to use the two alternatives under evaluation for completing their task assignments. In a first phase of the experiment, half of the students were asked to complete tasks by making use of the RDBMS report generation capabilities and the other half of the students were required to complete their assignments by making use of the prototype of our approach. In a second phase of the experiment, all students were asked to get all tasks solved again but using the alternative software. *Time consumption* was computed for each task / each student. The amount of computing resources used in each case was computed as well. It was a requirement for the students that the resulting data views using one or another method should retrieve the same data.

Table 3 and table 4 illustrate the mean value of *Time consumption* for each task and its standard deviation. Values are expressed in seconds. Variables v and r correspond to the result obtained by students in their first attempt to solve tasks using *ViewOnto* and *RDBS*, respectively. Variables v' and r' are the same variables v and r but during the execution of the experiment in its second phase.

As deducible from the tabular data, the students obtained the solution faster when using our approach in 90% of cases in both phases of the experiment. Making an analysis of *Time consumption*, we conclude that the computing capabilities required for making use of our software prototype to changing environments are irrelevant compared to those required when extensions to IDE Eclipse were used to update the equivalent data report and/or user interface. The performance of the software prototype highlights the relevance of the chosen software architecture (100% JavaScript).

Table 3: Experiment results in phase one

Task	v	$\sigma(v)$	r	$\sigma(r)$
T1	75	8	92	18
T2	58	22	75	4
T3	44	7	83	10
T4	44	16	122	12
T5	49	16	135	8
T6	43	10	41	2
T7	403	35	420	32
T8	445	24	905	49
T9	394	42	574	32
T10	51	9	70	8

Table 4: Experiment results in phase two

Task	v'	$\sigma(v')$	r'	$\sigma(r')$
T1	74	4	72	15
T2	56	4	71	5
T3	43	3	63	9
T4	42	3	124	14
T5	47	5	119	6
T6	41	2	42	2
T7	394	29	361	30
T8	445	23	812	47
T9	386	39	542	36
T10	51	9	60	7

It seems that costs of maintaining and keeping up to date the data model decrease. The experiment results actually show that working with *ViewOnto* is less time consuming than doing the corresponding updates in a traditional RDBMS (v and v' values are smaller than the corresponding r and r' values in 90% of chances). Further, when comparing the mean values of r to r' and v to v', we corroborate that *Time consumption* was considerably reduced when using *ViewOnto* in the first phase of the experiment (working with the *RDBMS* after solving tasks with *ViewOnto* was 35% less time consuming than working with the *RDBMS* in the first phase of the experiment). Further experimentation is required but results indicate that our proposal actually facilitates the understanding of the meaning of data in the context it is used.

6 Conclusions and Future Work

In our approach, data are stored in RDBs with their proven track record of scalability, efficient storage, optimized query execution, and reliability. However, compared to the relational data model, OWL demonstrated sounder capabilities of expressivity. By describing RDBs in *ViewOnto*, our proposal allows users to focus on data retrieving taking into account the explicit semantic formal definition of data views.

A prototype of a general tool for data retrieving was implemented and used for testing. By using *ViewOnto*, the flexibility of the RDBMS of the study case increased because no recoding of the application was needed. The maintenance costs decreased because user interaction with the prototype and changes in the A-Box of *ViewOnto* took less time than the traditional recoding in the application. Further, it seems that having an explicit formal description of data views facilitates the understanding of the users, improving their capabilities for managing the system in the traditional way. The applicability of the proposal in teaching database design has been envisioned.

Our future work will focus on the improvement of data retrieving processes based on the inference of new information using *ViewOnto* and continuing the study of the usability of semantics-based data recovery systems.

References

[1] Y. Lin, Ch. Cole and K. Dalkir , The relationship between perceived value and information source use during KM strategic decision making: A study of 17 Chinese business managers, Information Processing & Management, Vol. 50, No. 1, 156-174, (2014).

[2] C. Bolchini, E. Quintarelli, L. Tanca, CARVE: Context-aware automatic view definition over relational databases, Information Systems, Vol. 38, No. 1 (2013).

[3] T. Niemi, S. Toivonen, M. Niinimki and J. Nummenmaa,Ontologies with Semantic Web/grid in data integration for OLAP. International Journal on Semantic Web and Information Systems, Special Issue on Semantic Web and Data Warehousing, Vol.3, No. 4 (2007).

[4] Ch. Chen, G. Zhao, Y. Yu, Multiple views system to support awareness for cooperative design, Computer-Aided Design, Vol. 63, 39-51 (2015), doi:10.1016/j.cad.2015.01.001.

[5] B. Motik, I. Horrocks and U. Sattler, Bridging the gap between OWL and relational databases, Web Semantics. Vol. 2, No. 2, 74-89 (2009).

[6] A. Chebotko, S. Lub, X. Fei and F. Fotouhi, RDFProv: A relational RDF store for querying and managing scientific workflow provenance , Data & Knowledge Engineering, Vol. 69, No. 8, 836-865 (2010).

[7] H. Sun and Y. Fan, Semantic Extraction for Multi-Enterprise Business Collaboration, Tsinghua Science & Technology, Vol. 14, No. 2, 196-205 (2009).

[8] I. Horrocks, Peter F. Patel-Schneider and Frank van Harmelen, From SHIQ and RDF to OWL: The Making of a Web Ontology Language, Web Semantics, Vol. 1, No. 1, 7-26 (2003).

[9] H. Agus-Santoso, S. Cheng-Haw and Z. Abdul-Mehdi, Ontology extraction from relational database: Concept hierarchy as background knowledge, Knowledge-based Systems, Vol. 24, No. 3, 457-464 (2011).

[10] K. Čerāns, G. Būmans. RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language, Proceeding of the 2011 Conference on Databases and Information Systems, Amsterdam, The Netherlands, IOS Press, 139-152 (2010).

[11] K. Munir, M. Odeh and R. McClatchey, Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL, Knowledge-based Systems **35**, 144-159 (2012).

[12] M. Al-Sudairy and T. Vasista, Semantic data integration approaches for e-governances. Web & Semantic Technology (IJWesT), Vol.2, No.1. (2011).

[13] M. Hert, G. Reif and H.C. Gall, Updating relational data via SPARQL/update, EDBT '10: Proceedings of the EDBT/ICDT Workshops, ACM, New York, NY, USA (2010).

[14] Ch. Dichev and D. Dicheva, View-based Semantic Search and Browsing, Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 919-925 (2006).

[15] J. Hong, E. Suh and S. Kim, Context-aware systems: A literature review and classification, Expert Systems with Applications, Vol. 36, No. 4, 8509-8522 (2009).

[16] D. Namiot, Context-Aware Browsing – A Practical Approach, Proceedings of the Sixth International Conference on Next Generation Mobile Applications, Services and Technologies(NGMAST), International Conference on. IEEE, 18-23 (2012).

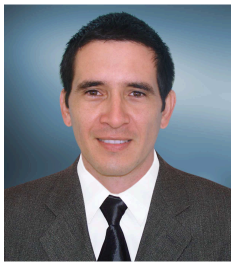
[17] M. Hert, G. Reif and H.Gall, A comparison of RDB-to-RDF mapping languages, Proceedings of the 7th International Conference on Semantic Systems ACM, 25-32 (2011).

[18] Z. Wu, H. Chen and X. Jiang, Semantic-based Database Integration for Traditional Chinese Medicine, Modern Computational Approaches to Traditional Chinese Medicine, 199-212 (2012).

[19] K. Vavliakis, T. Grollios and P. Mitkas, RDOTE - Publishing Relational Databases into the Semantic Web, Systems and Software, Vol. 86, No. 1, 89-99 (2013).

[20] F. Song, G. Zacharewicz and D. Chen, An ontology-driven framework towards building enterprise semantic information

layer, Advanced Engineering Informatics, Vol. 27, No. 1, 38-50 (2013).
[21] T. Barsalou, W. Sujansky, L.A. Herzenberg and G. Wiederhold, Management of complex immunogenetics information using an enhanced relational model, Computers and Biomedical Research **24**, 476-498 (1991).
[22] A. Guido and R. Paiano, Semantic integration of information systems, Computer Networks & Communications, Vol. 2, No. 1 (2010).
[23] M. Aufaure, R. Chiky, O. Cur, H. Khrouf, G. Kepeklian, From Business Intelligence to semantic data stream management, Future Generation Computer Systems (2015), doi: 10.1016/j.future.2015.11.015.
[24] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation (2013).
[25] G. Antoniou, O. Corcho and K. Aberer, Semantic Data Management, Report from Dagstuhl Seminar (Dagstuhl Seminar 12171) **2**, 39-65 (2012).
[26] F. Baader, D. Calvanese and D. McGuinness, The Description Logic Handbook. Theory, Implementation and Application, 2nd edition. Cambridge University Press (2010).
[27] Y. Ponce, M. Pérez, F. Fernández and J. Nummenmaa, Ontology-based Dynamic Building of Relational Data Views, Procedings of the Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support, Atlantis Press, **51** (2013), doi:10.2991/.2013.46.
[28] F. Fernández-Peña, R. Acosta-Sánchez, Y. Ponce-Toste. ViewOnto: modelo conceptual para la generación automática de vistas de datos, Ciencias de la Información, Vol. 46, No. 1, 19-25 (2015).

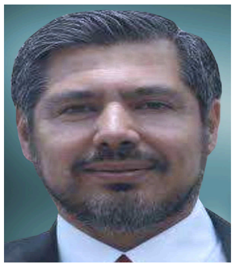


Félix Fernández is Professor of Software Engineering at the Universidad Técnica de Ambato. He received his PhD in Software Engineering at the Polytechnic University of Havana in 2007. His main research interests include: data management, end-user data analysis, semantic web, social networks and web programming.

Pilar Urrutia is Professor of Software Engineering at the Universidad Técnica de Ambato. She received her Master Degree in Information Systems in 2002, and her Master Degree in Data Networks and Telecommunications in 2006, both in the afore mentioned university. Her main research interests include: virtual e-learning objects, data networking, social networks and end-user data analysis.

René Cañete is Professor of Mobile Platform Programming and Digital Electronic at the Universidad Tecnológica Israel. He received his PhD in Software Engineering at the Polytechnic University of Havana in 2010. His main research interests include: data management, end-user data analysis, semantic web, social networks, web programming and mobile programming.

Rolando Acosta is an Algebra and Artificial Intelligence professor at the Polytechnic University of Havana, Cuba. He received his MSc Degree on Applied Informatics from the Polytechnic University of Havana in 2011. His main research interests include: artificial intelligence, data mining and semantic web.



Cornelio Yañez received his BS degree in Physics and Mathematics at the Escuela Superior de Física y Matemáticas at IPN in 1989; the MSc degree in Computing Engineering at CINTEC-IPN in 1995, and the PhD at the Centro de Investigación en Computación (CIC-IPN) in Mexico City in 2002, receiving the Lázaro Cárdenas award 2002. Currently, he is a Titular Professor at the CIC-IPN. His research interests include Associative Memories, Mathematical Morfology and Neural Networks.

Jyrki Nummenmaa is a professor of Computer Science at the University of Tampere, Finland, where he is the head of the research center CIS: The Tampere Research Center for Information and Systems. He received his PhD in Computer Science at the University of Tampere on 1995. His main research interests are: end-user data analysis, traffic data analysis, data management, software development, and algorithms.