International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 04 - Issue 02 || February 2018 || PP. 12-20

A Comparison of Different Perspectives for the Design of Ontology-Database View

Amer Ibrahim
Assistant professor
Department of Informatics Systems, Emirate College for Management and Information Technology
Dubai, United Arab Emirates

Abstract: An ontology is a complex information object. It can contain millions of concepts in complex relationships. This creates the need to establish a small ontology as an alternative version of a base ontology. This paper aims to present the different mechanism of creating an ontology views. Ontology views are argued to offer a solution to large and unwieldy ontologies, providing custom views that can act as ontologies in their own right. The survey is structured such as to give a brief overview of what ontology views are, what work has been done to generate ontology views from both a query or algorithmic perspective.

1. Introduction

The latest version of the web, Semantic Web (SW), and related technologies have the potential to change the current web, and promise to make it an interesting resource for both users and researchers. Ontologies form the core of the SW, and are used to explicitly represent our conceptualizations. They also play a central role in SW, due to the fact that they describe semantic information relationships. Ontology is a content theory about the kinds of objects, properties of objects, and relations between objects, that are possible in a specified domain of knowledge [4]. Currently, ontologies are large or tend to grow larger than their original size, causing problems such as: being too large and potentially not being understood due to their size. There is, thus, a need to investigate the use of ontology views in SW applications. An ontology view extracts relevant segments from large ontologies, in order to increase tractability for both humans and computers. An ontology view mechanism allows ontologies to be accessed and manipulated via a simplified "view", and allows the ontology to be used in conjunction with an instance store. To define an ontology view mechanism it is also necessary to determine the view language, which chooses data required by the user from the ontology and allows the view to be, easily created with respect to the user's aims. Nowadays there are many systems in place for this purpose; in our research, we have observed that authors of view mechanisms have brought their experience, from other fields and combined it with the current ontology view, to produce a new one. The aim of our study is to provide the motivation and benefits of view, an overview of existent mechanisms for querying and creating ontology views, present their current shortcomings, and define the requirements of new ones.

Following this introduction, Section 2 presents Ontology view motivations and benefits, and their similarity to database view. Discussion of some of the mechanisms that have been used for ontology view creation is presented in Section 3. Finally, we finish with our conclusions in section 4, followed by the references section.

2. Ontology Views

Current research presents various approaches for view extraction. But these works are constrained by: (a) available ontology representation language or notation; (b) the lack of standardized storage models (and structure) for storing and querying ontology bases; (c) the lack of standardized ontology query languages; and (d) the lack of efficient tools for view extraction. Thus, there are many critical points for ontology view construction that are lacking, and therefore a great need for more work to be carried out, in order to build and clarify their roles.

2.1 Ontology View Motivations

There is an urgent need for ontology views mechanisms, because ontologies tend to grow larger than their original size, introducing such problems as: (i) being too large to be utilized in their original scale by potential applications; and (ii) potentially not being understood by the community due to their size. As previously mentioned, an ontology base tends to grow larger with usage. Therefore, there exists the need to extract a portion (sub-section or sub-ontology) of the main ontology that is of interest to the user for a given task. This prevents the over-utilization of valuable computing resources and saves users' time, when carrying out their tasks, while also providing other benefits such as privacy, security and efficient access. Another reason for localized sub-ontology is the processing time involved in executing semantic queries over large-scale ontology bases (for example, HDC, the PO).

2.2 Ontology View Benefits

The motivation for creating views has changed in recent years, from a great amount of research carried out by both researchers and industry, to an improvement in the quality of the design, construction and performance of ontology views. From the following list, it is clear that the applications and benefits of views have extended beyond their originally intended aims (Data Extraction and Elaboration): (a) user access and user access control (UAC) applications, (b) defining user perspectives/profiles, (c) designing data perspectives, (d) dimensional data modeling, (e) providing improved performance and logical abstraction (materialized views) in data warehouse/OLAP and Web-data cache environments, (f) Web portals and profiles, and (g) SW paradigms [21] for sub-ontology or ontology views.

Ontology views promise to: (a) provide a manageable portion of a larger ontology for the localized applications and users, (b) enable precise extraction of sub-ontologies of a larger ontology that commit to the main ontology, (c) enable localized customization and usage of the portion of a larger ontology and (d) enable interoperability between large ontology bases and applications.

2.3 Antecedents: Database View

The very large volume of data is a significant problem for ontology data management in the Semantic Web environment. In addition, the continual increase in web resources, which cause frequent updates of ontology data. Thus, there is a need to investigate utilization of an ontology view, as an alternative version of ontology. Many of the existing works have tried to solve these problems, using methods based on database; therefore we present a brief discussion of database view and its similarity to ontology view.

View expression was first introduced in Relational database, and it has been widely used in many database applications. This is therefore a well-defined and structurally sound area, which due to the quantity of works produced, provides us with many standards and solutions to view problems. It also provides us a base, upon which we can build or extend our work to extract an ontology view, instead of re-inventing the wheel.

Database and ontology serve to structure the vast amount of information that is available at a given point in time. However, they have many differences: in database, schemas are defined in one level of abstraction (logical or schemata level), and view is defined as part of the external schema, in ontology, conversely, the schema is defined at varying levels of abstraction and instances may co-exist among schemas to convey information, concepts or relationships between two concepts to users. Another difference is that database is well defined as an established standard, while ontology tends to have different standards and models. Accordingly, Rajugan, Chang and Dillon have addressed the following points about ontology views [16]:

1. Unlike database or semi-structured views, sub-ontologies are not just an extracted portion of the main ontology base, but a collection of concepts, relationships and concerns that itself is a new interpretation of the base ontology.
2. The meaningful representation of such sub-ontologies, that are easily understood by humans, as well as easily transformed to machine (or user-application) readable notations, that are at a level of abstraction, that is capable of interpreting, querying and processing of concepts, relationships and constraints.
3. The complementary issues associated with sub-ontologies, such as view maintenance, versioning and materialization, which are synonymous with database views, but deserve detailed studies of their own, in the context of ontologies.
4. The issue of meaningful, yet efficient extraction of sub-ontologies from distributed base ontologies.

3. Ontology Views Defintion Solution

Part of our own work has involved the definition of an ontology view, motivation, benefits and the similarities to database. In this section we discuss some current ontology view mechanisms that, in spite of their diversity and the different approaches they use, aim to provide the ontology view as the most appropriate form, from the point of view of author. However, we must first refer to the requirements necessary to the Ontology view [20]:

1. The structure of views must correspond to the structure of data.
2. The semantics of views must be specified by ontology, and should be embedded in the respective inheritance hierarchies according to their semantics by classification. This classification must be part of every view definition.

3. The view's definition of an ontology is an ontology. This means the requirements of ontology languages must be included in the view languages [7].
4. It is imperative to maintain the quality of the view (data) generated, without loss of semantics.

3.1 Classification of View works

Recently much research has focused on presenting the advantages of ontology view, and these works can be classified in different ways. David Taniar and Johanna Wenny Rahayu have grouped them into four categories, namely: (a) classical (or relational) views, (b) object-oriented view models, (c) semi-structured (namely XML) view models, and (d) view models for SW [15].
Recent works on ontology views have also been classified by another method, which groups aspects according to the following fields [8]:

1. Ontology definition languages as RDF/S, OWL.
2. Ontology Query languages as RQL [9], RVL [11], SPARQL.
3. Semantic integration of ontologies, and graphical ontology editors as OntoEdit, Protégé 2000, ODE3.
4. The collaborative ontologies environments, such as Ontolingua Sever [3], OntoEdit and CODE [6], whose architectures are based on a centralized system with a server of ontologies.
5. Distributed environment without a central server, such as Wiki@nt [1] and OntoPathView [8], in the case of Wiki@nt the system capabilities for communication between developers are very limited.

Based on the mechanisms we have studied, we can add a new classification depending on the extraction process, consisting of two groups:
Query-Based Approaches: the languages for querying the base ontology which return ontology partitions in response to user needs.

Non-Query Approaches: use algorithms to extract the concepts and the relevant relations to build the new ontology (view ontology).
In the following section we present some examples of works for each group in our new classification.

3.2 Query-Based Approaches

Here we present various frameworks that enable users to design customized ontology views, and demonstrate that the views are the correct mechanism, for enhancing the usability of ontologies. These models depend on an ontology view language, which meets the needs of the user, and ensures that the specifications are achieved. Consequently, recent research on SW has mainly focused on aspects related to querying and viewing the ontologies. Thus, the problems that make querying SW Resources difficult have been grouped as follows [10]:

1. User Perspective: Users who are interested in finding resources formulate queries related to a particular ontology.
2. Application Integration: In the current state of SW, ontologies are developed from scratch, and therefore many ontologies describing the same domain exist. It is also widely believed that throughout the future development of SW, this ontology heterogeneity will remain and multiple ontologies for any particular domain will coexist.
3. Performance Overhead: In many SW domains large or very large data sets exists. Queries can produce a considerable performance overhead. Problem-specific views of the resources could potentially minimize this problem.
4. Lack of formal definitions: In the domain of e-learning, brokerage platforms (such as UNIVERSAL) and RDF-based peer-to-peer networks (such as Edutella), have been developed, which act as common mediators for accessing multiple data sources. Although various standardized vocabularies for metadata were developed in the last decade (e.g. LOM, SCORM), they are mostly informally defined and do not allow deeper reasoning.

In this section we present a brief description and the properties of various view systems, such as: CLOVE, RVL, Ontolingua Server and TRIPLE.

Clove

CLOVE is a Constraint Language for Ontology View Environments [19]. Is a start toward creating an application view from a foundational ontology. The authors defined a language that specifies inclusions or exclusions based on simple constraints, and present a mechanism for creating and querying view. The authors have focused on the dual nature of views as classes in the ontology and contexts to interpret new queries; they have focused on the systematic description and management of views as first-class objects in ontologies. CLOVE takes this duality into account and allows users to both create and query views. One of the most important design principles of CLOVE is that all users with access to the ontology should be able to create views.

RVL

RVL [11] is the First Declarative Language for creating virtual RDF/S resource descriptions and schemata. It exploits some RQL. Research into RVL has determined many important issues that must be dealt with, such as the composition of queries formulated against a view with the definition of the view in order to produce queries against the original RDF/S data that can be actually evaluated, checking the consistency of view definitions, and checking whether the graph they produce satisfies the constraints of user model. This allows views to be customized on-the-fly for specific applications' requirements. They however also side-step the ontology updating problem by only creating virtual views. Their views are merely a collection of pointers to the actual concepts, and are discarded after they have served their purpose.

Ontolingua Server

The Ontolingua uses the representation languages, Ontolingua Frame Ontology and KIF, which are wide spectrum language capable of representing fine features of concepts. Ontolingua Server [5] extends the original language in two ways: (1) it provides explicit support for building ontological modules that can be assembled, extended, and refined in a new ontology; (2) it separates ontologies presentation and representation. The main benefit of extending Ontolingua Language and its presentation in the Ontolingua Server is that axioms which do not fit into the frame language are allowed, meaning there is no restriction on expressiveness. The Ontolingua server supports ontology inclusion and circular dependencies. Its consistency-check capability, however, is restricted to the functions similar to database schema checking for instance, all slots, slot values, facets and facets values are checked to make sure that they conform to the constraints that they apply. This is extremely important for an ontology development environment. On other hand a clear separation is made between its simple formal semantics and the input/output properties of the system that uses it.

TRIPLE View

This mechanism uses Triple Language which is a rule language based on Horn logic [10]. It includes many basic features from F-Logic, and presents other concepts to view extraction which separate two types of ontology: source (base) and target (view) ontologies, and the mapping mechanism. This approach allows clients to formulate queries only in the target ontologies. As this technique uses Triple language then the target ontology can be presented in a different ontology definition language. TRIPLE View offers a solution to the ontology view evolution problem because when the source evolves, the changes must update the mapping between the source and base, and thus the users are not affected by changes in the source ontology.

3.3 Non-Query Approaches

Another approach to extract the ontology view is based on the use of algorithms to replace the base ontology with a smaller one, providing better performance and responding to user needs without modifying the main ontology structure. These methods vary in terms of how to submit ontology view, where some approaches offer a materialized view and others offer a virtual view or extract a view from one or many ontologies. Currently ontologies are widely used, and thus the proposed approaches for ontology view extraction will play an important role in improving the efficiency of ontology view. We subsequently provide a brief description of some of these methods.

Move

Materialized Ontology View Extractor [2, 22] is proposed as a distributed architecture for the extraction/optimization of sub-ontology from a large-scale base ontology. This distributed mechanism has the possibility of making the process faster and the extraction easer. It is a generic system that can theoretically be adapted to work with any ontology format. The system extracts a sub-ontology based on a user's labeling of which ontology terms to include and which to exclude. It also has the ability to optimize an extract based upon a set of user selectable optimization schemes. These extracts can be further restricted by enforcing a set of additional constraint.

Traversal View

Traversal view [12] enables users to extract self-contained portions of an ontology related to a particular concept or a set of concepts. It is based on user concepts as a starting point and follows links to build the user view, building the new view from the extracted concept list without modifying the base ontology structure. The extraction methodology focuses on traversal directives, which define how the ontology links should be traversed. They also introduce the concept of boundary classes around the edges of an extract. They used a developed version of PromptDiff algorithm resolves the view evolution problem and ensured the consistency between user view and base ontology.

Layered View Model (LVM) for Ontologies

This is a type of transformation built on three operations: Extraction, Elaboration and Extension. Extraction is defined as the extraction of part of the base ontology without any modifications, Elaboration is the provision of additional levels of detail to the extracted part and Extension is the addition of completely new elements. LVM presents a high level of abstraction (conceptual, logical or document levels) and maintains the data quality while at the same time preserving the semantics. The authors have affirmed that the ontology view should be usable as a standalone ontology for user applications. This work has been presented as a standard mechanism because it focuses on topics such as security, privacy, performance, extracting materialized views and providing ontology management and maintenance [20].

Multiple Views by RDFS

This model discusses how to exploit RDFS and use its features to provide a multi view in an ontology [18]. It is theoretical work based on RDF schema and RDF documents, where Individual RDF documents are validated against RDF schemas. RDF documents consist of descriptions, which in turn consist of statements. In this work they have treated ontology as if it was a black box, and the main problem that has been presented is the division of concept into classes and properties, the answer proposed to this problem is the usage of an external ontology in addition to the RDF(S) or reformulating the properties, where An ontology independent of the domain-specific details of its usage is needed. There should be an "isolated basic backbone" of ontology that is independent of any case-specific details and it should be clear that RDF(S) alone does not fit together with this requirement.

Web Ontology Segmentation

This technique is based on extracting relevant segments started from user choice and including all the related hierarchy to a determined depth [22]. It is the only technique that is based explicitly on OWL; the segments extracted are as small as possible, normally up to one fifth the size of the original and are related to the filtering property and depth limitation, this approach aims to create an independent ontology instead of ontology view. It takes advantage of many ontology maintenance principles, such as normalization, upper-ontologies and rich property hierarchies all of which are taken into account to produce more relevant segments. Ontology segmentation techniques allow the relevant, self-standing custom ontologies required by users to be created quickly and easily, instead of having to rely on the initial authors' decomposition.

3.4 Comparison of ontology view approaches

Having discussed many mechanisms for view extraction in our classification, we subsequently present a comparison between the mechanisms of each group. We will focus on many criteria relevant to the studied field, which can be seen in Tables 1 and 2.

Comparison of Query-Based Approaches

To demonstrate the flexibility and integrity of the languages being reviewed, or their feasibility for use as a standard query language, we have included many characteristics and some of their properties and functionalities in our comparison. As some of these properties, such as data model, language of origin, closure and orthogonality, are language properties, therefore we cannot find them in the comparison of non-query approaches.

Ontology definition languages: represents the definition language of the base ontology RDF or OWL, where RDF is considered to be the most relevant standard for data representation and exchange on the Semantic Web. OWL is a promising ontology language which is intended to provide a formal description of concepts, terms and relationships within a given knowledge domain.

Data model: The RDF model is composed of triples (subject, predicate and object); OWL is a subset of RDF. Moreover, the triples are related to form graphs; in table (1) we can see that Triple is the only query language to use a triple data model, while all other languages use a graph data model, which is useful for complex queries.

Language of origin: from studying these mechanisms we can see many models for ontology querying; Triple relies on a logic-based model, while the other languages follow either an SQL or an OQL approach. OQL differs from SQL in that[1]: (1) OQL supports object referencing within tables. Objects can be nested within objects, (2) not all SQL keywords are supported within OQL, (3) OQL can perform mathematical computations within OQL statements.

View Language: refers to the view language such as an extension to another ontology query language (for example, RVL (which extends RQL)) or if it is a separate language.
Regarding the property of **Closure,** this presents the possibility to support nested queries o when a function is defined within another function. Nearly all these languages support the nested and composition queries.

Orthogonality: is an Input/output type, where the output could be the input of another query, and any kind of data can also be permitted as the input and output of queries.

Multi Base Ontology: some of these languages can extract view from more than one base ontology, which provides us with the possibility to extract a view that responds to user needs from several ontologies, increasing the Evolution problem at the same time.

Materialized view: is used to increase the speed of queries in a large amount of data and improve query performance by recalculating expensive join and aggregation operations. It is clear that the majority of languages presented in this paper support this criteria.
Looking at the criteria of **View Evolution,** it can be observed that OntoLingua supports this characteristic in an integrated environment, while the other languages suffer from the consistency problem when the base ontology evolves or a view modifies. Here, we can clearly see the lack of approaches that ensure that views are shared and safely used in a distributed environment.

Table 1 comparison of Query-Based Approaches

Language / Criteria	RVL	Triple	Ontolingua	Clove
Ontology Definition Language	RDF/S	RDF/S	RDF/S	OWL
Data Model	Graph	Triple	Graph	Graph
Language of Origin	OQL like	F-logic like	KIF like	OQL like
View Language	RQL Extension	Triple	KIF Extension	OWL Extension
Closure	Yes	No	Yes	Yes
Multi Base Ontology	No	Yes	Yes	No
Orthogonality of Input/ Output Data	Yes	No	Yes	Yes
Materialized View	Virtual View	Yes	Yes	Yes
View Evolution	No	Yes	Yes	No

[1] http://en.wikipedia.org/wiki/Object_Query_Language.

Comparison of Non-Query-Based Approaches

From the discussion of the mechanisms presented in this paper we can see the growth role and the benefits of the ontology view as a necessary solution for using a large and complex ontology. As we have mentioned, the selected characteristics demonstrate the robust structure and the ability of these mechanisms to fulfill their role as a view extractor. In the following table (2) we present some criteria regarding the structure and functionality of the studied mechanisms. Nearly all these properties have been included in the previous comparison, therefore we will present the new ones.

Approach Name: During our research we found several terms such as sub-ontology, personal ontology, segment and partition which all have the same meaning and refer to ontology view.

It is implemented: It is important to mention that the RDFS View is the only one that is unimplemented, as can be seen in Table (2), presenting instead a theoretical approach, while the other approaches are implemented or integrated with other projects.

Ontology Example: Some of these mechanisms have used a standard ontology, such as Web Ontology Segmentation that used GALEN.

Table 2 comparison of Non-Query Approaches

Mechanism / Criteria	MOVE	Traversal View	LVM	RDFS Views	Web Ontology Segmentation
Ontology Definition Language	RDFS/ OWL	RDFS	RDFS / OWL	RDFS	OWL
Approach Name	Sub Ontology	Sub Set	View	View	Segment
Multi Base Ontology	No	No	No	No	No
Materialized View	Yes	Yes	Yes	No	Yes
It is Implemented	Yes	Yes	Yes	No	Yes
View Evolution	No	Yes	No	Yes	Yes
Ontology Example		FMA			GALEN

4. Conclusion

This work has focused on the huge size of an ontology, and its influence on the usability of the base ontology, and discussed the effectiveness of ontology view as the solution to resolving this problem. We consequently presented a comprehensive review of ontology view development and recent mechanisms that we have classified into two groups. Some of the aforementioned approaches are based on query language or extend other languages to construct an ontology view, while others use algorithms that extract a view independently of ontology query languages. Subsequently, we presented the difficulties of extracting an ontology view, and the

requirements that should be included in the view structure. Table (3) shows the studied mechanisms and the points they have in common according to our two groups. From this table it can be seen that some of the points currently lacking in the ontology view field, build an ontology view depending on the ontology definition languages. Some of the mechanisms presented can be easily adapted to support RDFS and OWL, but are built to use with one or the other. Examples of this are MOVE and Traversal. OWL and RDFS are W3C standard, but OWL is a promising ontology language which is intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain, and provides very flexible and good expressive powers. Another point that is lacking is the evolution problem, which causes an inconsistency between the base ontology and the view. Some of the mechanisms presented provide a solution to evolution problems, such as the building view by Traversal, but in other works the solution of this problem has been proposed as future work.
We emphasize that the view should not be limited to a query returning part of the ontology. Instead, it should be extended to the restructuring of class and property hierarchies, allowing the creation of new resources and property values, and even new classes and types of property, as well as maintaining the hierarchical structure without semantic loss.

Table 3. Ontology view comparison

Criteria / Mechanism	**Ontology Definition Languages**	**Multi-Base Ontology**	**Materialized View**	**View Evolution**	**Implemented**
Query-Based Approaches					
RVL	RDF/S	No	Virtual View	No	Yes
Triple	RDF/S	Yes	Yes	Yes	Yes
OntoLingua	RDF/S	Yes	Yes	Yes	Yes
Clove	OWL	No	Yes	No	Yes
Non-Query Approaches					
Move	RDF/S, OWL	No	Yes	No	Yes
Traversal view	RDF/S, OWL	No	Yes	Yes	Yes
LVM	RDF/S, OWL	No	Yes	No	Yes
RDFS View	RDF/S	No	Virtual View	Yes	No
Web Ontology Segmentation	OWL	No	Yes	No	Yes

5. REFERENCES

[1]. J. Bao and V.G. Honavar, "Collaborative Ontology Building with Wiki@nt" Third International Workshop, on Evaluation of Ontology Building Tools, Hiroshima 2004.

[2]. M. Bhatt, A. Flahive, C. Wouters, J.W. Rahayu and D. Taniar, "MOVE: A Distributed Framework for Materialized Ontology View Extraction", Algorithmica, No. 45, pp. 457–481, Springer, 2006.

[3]. R. Berlanga, A. Scheppler, M. J. Aramburu Cabo, I. Sanz and R. Danger, "OntoPath: A Query Language for Ontologies", IX Conference of Software Engineering and Data Bases, Malaga, 2004.

[4]. B. Chandrasekaran, J.R. Josephson and V.R. Benjamins, "What are ontologies, and why do we need them?", IEEE Intelligent Systems, pp. 20-26, 1999.

[5]. A. Farquhar, R. Fikes and J. Rice, "The Ontolingua server: a tool for collaborative ontology construction", International Journal of Human-Computer Studies, vol. 46, no. 6, 1997.

[6]. P. Hayes, R. Saavedra and T. Reichherzer, "A Collaborative Development Environment for Ontologies (CODE)", Semantic Integration Workshop (ISWC), Sanibel Island, Florida, October 2003.

[7]. J. Heflin, R. Volz and J. Dale, "Requirements for a Web Ontology Language", W3C Working Draft, http://www.w3c.org/TR/webont-req, March, 2002.

[8]. E. Jimenez, R. Berlanga, I. Sanz, M. J. Arsmburu and R. Danger, "OntoPathView: A Simple View Definition Language for the Collaborative Development of Ontologies", Universidad Jaume I de Castellón, 2005.

[9]. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis and M. Scholl," RQL: A Declarative Query Language for RDF", International World Wide Web Conference Honolulu, Hawaii, USA, May 7-11, 2002.

[10]. Z. Miklos, G. Neumann, U. Zdun and M. Sintek, "Querying semantic web resources using triple views", International Semantic Web Conference (ISWC), Sundial Resort, Sanibel Island, Florida, USA, October 2003.

[11]. A. Magkanaraki, V. Tannen, V. Christophides and D. Plexousakis, "Viewing the Semantic Web through RVL Lenses", International Semantic Web Conference ISWC, 2003.

[12]. N. F. Noy and M. A. Musen, "Specifying ontology views by traversal", In Proceedings of the Third International Semantic Web Conference (ISWC2004), 2004.

[13]. I. Ohmukai and H. Takeda, "Metadata-driven Personal Knowledge Publishing", Proceedings of the Third International Semantic Web Conference (ISWC2004), 2004.

[14]. R. Rajugan, "A Layered View Model for XML with Conceptual and Logical Extension, and Its Applications", PhD Thesis, University of Technology, Sydney (UTS), Australia, Sydney, 2006.

[15]. R. Rajugan, E. Chang, T.S. Dillon, L. Feng and C. Wouters, "MODELING ONTOLOGY VIEWS: An Abstract View Model for Semantic Web", Octubre 11, 2005.

[16]. R. Rajugan, E. Chang and T. S. Dillon, "Sub-Ontologies and Ontology Views: A Theoretical Perspective", International Journal of Metadata, Semantics and Ontologies, vol. 2, pp. 94-111, 2007.

[17]. R. Volz, D. Oberle and R. Studer, "Implementing views for light-weight web ontologies", The Seventh International Database Engineering and Applications Symposium (IDEAS'03), July 16-18, 2003, Hong Kong, pp. 160-170, IEEE Computer Society, 2003.

[18]. S. Toivonen, "Using RDF(S) to provide multiple views into a single ontology", Semantic Web Workshop 2001.

[19]. R. Uceda-Sosa, C. Chen and K. Claypool, "CLOVE: A Framework to Design Ontology Views", ER 2004, LNCS 3288, pp. 844–849, Springer, 2004.

[20]. R. Volz, D. Oberle and R. Studer, "Views for light-weight web ontologies", In ACM Symposium on Applied Computing (SAC), Orlando, Florida, Pages 1168–1173, March 2003.

[21]. C. Wouters, T. S. Dillon, J. W. Rahayu, E. Chang and R. Meersman, "A Practical Approach to the Derivation of a Materialized Ontology View", In Web Information Systems, D. Taniar and W. Rahayu. Eds. USA: Idea Group Publishing, pp. 191-226, 2004.

[22]. J. Seidenberg and A. Rector, "Web Ontology Segmentation: Analysis, Classification and Use". In 15th International World Wide Web Conference, Edinburgh, Scotland, 2006.