XML-BASED RDF QUERY LANGUAGE (XRQL) AND ITS IMPLEMENTATION

Norihiro Ishikawa, Takeshi Kato, Hiromitsu Sumino
NTT DoCoMo Inc, 3-5 Hikarino-oka, Yokosuka, Kanagawa, Japan

Johan Hjelm
Ericsson Research, Torshamnsgatan 23, Kista SE-16480, Stockholm, Sweden

Kazuhiro Miyatsu
Ericsson Research Japan,1-4-14 Koraku, Bunkyo-ku Tokyo, Japan

Keywords: Mobile phone, RDF, XML, Query Language, Metadata, Semantic Web

Abstract: Resource Description Framework (RDF) is a language which represents information about resources. In order to search RDF resource descriptions, several RDF query languages such as RQL and SquishQL have been proposed. However, these RDF query languages do not use XML syntax and they have limited functionality. xRQL is proposed to solve these issues by defining an XML-based RDF query language with enhanced manipulations of the RDF metadata. xRQL is a logic language relying on a functional approach. It consists of an operator declaration, a RDF data description and a result description. Based on RDF graph data model, xRQL defines a graphical path expression with variables, which is similar to GOQL for describing RDF data. It also adopts the object-oriented model for *creation*, *modification* and *deletion* operations of RDF data. Users can define their favorite XML-compliant result descriptions by themselves, which is similar to XQuery. In addition, a set of RDF operations for RDF schema is defined to manipulate the class and property hierarchies in RDF schema. xRQL has been implemented as an RDF query language over a native RDF database management system. This paper also briefly describes the implementation status.

1 INTRODUCTION

The World Wide Web was originally built for human consumption. It contains such a large volume of information that it’s impossible to manage manually. The use of metadata to describe the information can achieve automatic processing. RDF (Resource Description Framework) (Ora, 1999, Dan, 2002) is a W3C recommendation for describing information objects and processing metadata.

For querying RDF resource description, several RDF query languages (G.Karvounarakis, 2002, Mike, 2002, L.Miller,2001, Flavius,2002, Joachim,2002, Edutella, 2002) have been developed. However, in the context of processing within an XML-based distributed database system, and with respect to an RDF and logic language, there are some problems to be solved:

- ***Non-XML Syntax***
  Since the existing RDF query languages use SQL-like syntax to describe an RDF query, a special parser other than an XML parser is required. SQL-like syntax has limitations in expressing a complicated RDF query, compared with XML-based syntax. It is also difficult to describe the hierarchical RDF data model with a plain text syntax language.
- ***Limitation of RDF Metadata Manipulations***
  The existing RDF query languages only support RDF metadata search operations. Other functions, such as *Creation*, *Modification* and *Deletion* of

RDF metadata should be considered to support metadata manipulation operations, as well as the similar functions are supported by SQL in a relational database system.

- ***Lack of Result Format Declaration***
The existing RDF query languages retrieve the search result in an unordered and ambiguous way. It takes more time for machine to understand and process the retrieval results. It will be effective and powerful if an expected result format is declared in the RDF query language.

As a solution to the above problems, we have proposed an XML-based RDF query language, named *xRQL*, to manipulate RDF resource description and RDF schemas. It relies on the RDF graph data model that captures the RDF modeling primitives and permits the interpretation of superimposed resource description. *xRQL* adopts the functionality of an XML query language (D. Chamberlin, 2001), several object-oriented query languages (ODMG,1998, Serge, 1989, Zeki, 1996, Flavius, 2002) and a graph query language (Jeonghee, 1999, R.Fikes, 2001). So it's expected to be a standard query language for manipulation of RDF resource description stored in the distributed RDF repositories.

Section 2 summarizes the requirements that an RDF query language should satisfy. In Section 3, we depict *xRQL* design concepts and *xRQL* syntax in details, and give some typical examples. In Section 4, a native RDF management system that supports *xRQL* is briefly introduced. Finally, we summarize our contributions and show our considerations in the future research in Section 5.

2 REQUIREMENTS OF AN RDF QUERY LANGUAGE

We summarize the general requirements of an RDF query language as follows.

- ***Machine Readability:*** RDF query language syntax should be human readable and machine understandable for processing RDF metadata easily and efficiently between the distributed heterogeneous metadata repositories.
- ***Support for RDF Data Model:*** RDF query language should support the RDF data model (e.g. resources, properties, values), which is similar to the object-oriented data model. This means RDF query language should support concepts in the object-oriented data model, such as class hierarchies and inheritance.
- ***RDF Schema Query:*** Besides the manipulation functions of the distributed heterogeneous metadata represented in RDF, the retrieval function of schema definitions for the vocabularies used in any given block of RDF data is needed. Since schema definitions are defined by using the RDF Schema (RDFS) Specification, which provide information about relationships between classes and properties which might be used in queries, RDF query language should be able to query RDF schemas which are implicit in any RDF data model.
- ***Inference:*** Because of the hierarchical vocabularies defined in RDFS specification, such as *subClassof* and *subPropertyof,* an RDF query language should support the operations of inferring "implicit" relationships between classes and properties.
- ***RDF Metadata Manipulation:*** In addition to basic search functions, RDF query language should support the operations of *creation*, *modification* and *deletion* of RDF metadata. These operations, not yet supported by the current query languages, will be attractive to the clients who want to construct their own RDF metadata repositories based on given RDF schemas.
- ***Definition of Search Result Format:*** In order to achieve the efficient processing of search results, the capability of defining search result format should be available in RDF query language. That will help the clients to retrieve the search results in their favorite formats.

3 XML-BASED RDF QUERY LANGUAGE: xRQL

xRQL is a logic language relying on a functional approach. It consists of an operator declaration, a RDF data description and a result description. Based on RDF graph data model, *xRQL* defines a graphical path expression with variables, in the same way as Graphical Object-Oriented Query Language (GOQL) (L.Sheng, 1999) for describing RDF data. It also adopts the object-oriented model (D.Chamberlin, 2001) for *creation*, *modification* and *deletion* operations of RDF data. Users can define their favorite XML-compliant result formats by themselves, in the same way as XQuery (ODMG, 1998). In addition, a set of RDF schema operations is defined to manipulate the class and property hierarchies.

3.1 Design Concept

We describe the design concepts of *xRQL* by using the following example of RDF resource for the web content. Figure 1 depicts the contents published on a certain web site. The lower part depicts the descriptions of two persons (r1 and r4) and two *content* of music and movie on the web (r2 and r5). The middle part depicts a schema for contents annotation. In this schema, property *create* is defined with domain "*person*" and range "*content*". The upper part shows the well-known RDF data model. Properties (e.g. *create*, *managed*) serve to represent attributes (or characteristics) of resources as well as relationships between resources. Furthermore, both classes and properties include hierarchical semantics. For example, class *music* (or *movie*) is a subclass of *content* while property *compose* (or *produce*) is a subproperty of *create*.

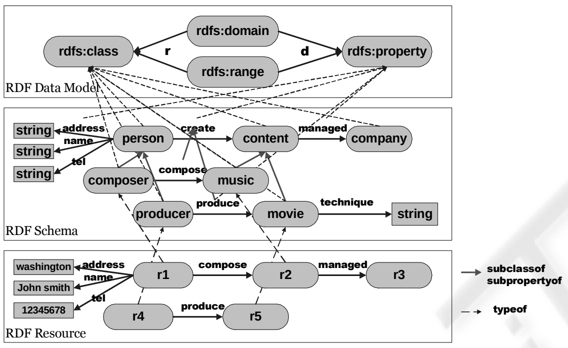

Figure 1: A schema of RDF resource for the web content

3.1.1 A Layered Graph Data Model

From our perspectives, RDF metadata architecture can be divided into three levels: *RDF Resource* level, *RDF Schema* level and original *RDF Data Model* level. (Figure 2) *RDF Resource* level defines labeled and directional graphs whose nodes are called resources or literals (such as r2 in Figure 1), and edges are called properties. *RDF Schema* level essentially defines vocabularies of labels for graph nodes and edges, such as class *person* and property *create*. *RDF Data Model* level includes the well-known RDF data model vocabularies, such as *rdfs:class* and *rdfs:property*, which describe vocabularies of the RDF schema. Each level can be viewed as an instantiation of the levels above it. More formally, those resources in *RDF Resource* level are seen as the instantiations of *RDF Schema* level. In the meantime schemas in *RDF Schema* level are seen as the instantiations of *RDF Data Model* level.

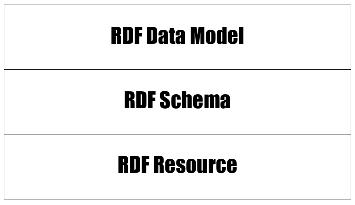

Figure 2: The layered data model

Then the relations between these three levels are ensured by a standard *type* interpretation. In Figure 1, the r1 in *RDF Resource* level has a directed *type* label pointing to *composer* in *RDF Schema* level whose *type* is *rdfs:class* of RDF Data Model level.
xRQL is designed based on this 3 layered data model.

3.1.2 RDF Resource Manipulation

According to the data model in section 3.1.1, we defined four basic operations to manipulate RDF resource under the constraint of *RDF Schema* level. These operations include *search*, *deletion*, *creation* and *modification*.

The *xRQL* queries essentially provide the means of manipulating RDF resource with the knowledge of the given RDF schema. For *search* operations, we proposed a path expression based on RDF graph data model. In compliance with RDF graph (i.e. nodes and edges) and human readability, the path expression is defined as a sequence of nodes and edges according to the given RDF schema, similar to GOQL, which is an extension of Object Query Language OQL (ODMG, 1998). A typical path expression is the following:

L = C1 {p1} C2 {p2} C3 …

where p1,…,pn are names of properties and C1,…,Cn are names of classes defined in the designated RDF schema. A data path is a sequence of nodes and edges (e.g. C1, p1, C2, p2, C3, …, pn, Cn), where the C*i*' s are classes and, for each *i*, there is an edge labeled p*i* between C*i+1* and C*i*. Implicit relation between C1 and Ci can be expressed as {p1}C2…Ci-1{pi-1} sequence.

Furthermore the variables can be defined in the path expression in order to combine the RDF schema with the associated RDF resource smoothly. For example, to retrieve the *person* resource that *create* certain *content* in Figure 1, the path expression could be written as:

person*: x {*create*} *content

Here, variable x represents the instance of a *person* class that will be retrieved. If there are subclasses defined under *person* class, the resource description belonging to these subclasses will also be retrieved. According to RDF schema defined in Figure 1, the instances of *composer* and *producer* subclasses will also be returned as the instances of the *person* class.

For RDF metadata manipulation (i.e. creation, deletion and modification operations), RDF resource descriptions are defined with object-oriented concepts, in the same way as OQL (ODMG, 1998). A resource that belongs to a certain class defined by RDF schema in the RDF repository is described as an object identified by its unique ID (i.e. URI) with properties and the associated values. In Figure 1, r1 is an object with *name*, *tel*, *address* and *compose* properties. The values of *name*, *tel* and *address* properties are defined as *string* while the value of *compose* points to another object r2. r1 object belongs to *composer* class and r2 object belongs to *music* class according to the RDF schema in the middle part of Figure 1. The RDF metadata manipulation is defined as the operations of objects which are constrained by the RDF schema. Figure 3 shows an object-oriented schema converted from the original RDF schema in Figure 1.

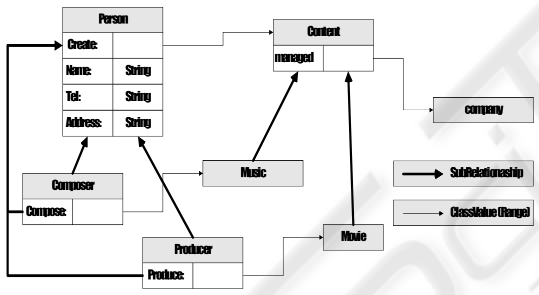

Figure 3: Object-oriented Schema

A *deletion* operation is used to remove the RDF resource from the RDF repository. This operation will remove an object completely. Not only the RDF resource itself and its properties, but also the associated statements will be removed. For example, "Delete the *company* r3 from figure 1" will remove the "r3" resource and all of its corresponding relationships completely. Values of property "managed" belonging to r2 and r5 will be changed to *Null* after the deletion operation.

A *creation* operation is defined to create a new RDF resource into RDF repository. As shown in Figure 3, each of the RDF resources can be represented as an object with a set of properties and associated values. The value of a property could be another object or *literal*. RDF resource inherits properties defined by its parent class. For example, the *composer* object inherits properties, such as *name, tel* and *address* from its parent class, namely *person* class.

A modification operation is defined to change the value of a designated property. If the value is another object, a relationship between the object that the property belongs to and the value object is established. As shown in Figure 4, this operation could also be used to merge or separate sub RDF graphs (e.g. right pattern in Figure 4).

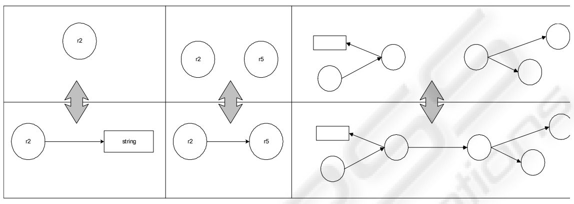

Figure 4: Modification operation based on object-oriented concepts

3.1.3 Syntax of RDF Query Language

According to the World Wide Web Consortium, the Extensible Markup language (XML) (T.Bray, 1998) is now spreading out as the standard language for machine understanding. Since RDF adopts the XML syntax to describe its data model, XML parser becomes a necessary tool for automatic machine processing of RDF resource description. In this approach, the use of XML syntax-based RDF query language which is consistent with RDF syntax will greatly lighten processing of the query analysis, compared with those queries defined in non-XML syntax, which often requires a specific parser.

3.1.4 Result Format Definition

The existing RDF query languages have no consideration of the result format definition in their syntaxes. For automatic processing of the retrieve result, users have to develop a tool to convert the result into machine-readable format. We think it is a bottleneck in the wide spread of RDF technologies. *xRQL* attempts to provide "format-able" result by providing an element for the users to flexibly define their favorite result format based on XML syntax, similar to *return* in XQuery. For example, in Figure 1 RDF schema, assuming that we have the variable x that has been defined in path expression: ***person:x{create}content*** retrieve with <personinfo>, the result format is as follows:

<result>

<personinfo> x </personinfo>

</result>

A set of person instances will be returned with tag <personinfo>. Using XSLT, this result could be easily converted into xHTML for displaying or further processing.

3.2 xRQL Syntax Definition

xRQL is a logic language relying on a functional approach. Its syntax is defined by XML syntax. *xRQL* syntax consists of three functional parts: operator declaration, RDF data description and result format description. The first part declares user's request operations. The second part defines the RDF resource or schemas using object-oriented concepts, path expression and condition declaration. The last part depicts user's favorite XML-based format of the result.

3.2.1 Operator Declaration

An operation declaration element is defined to declare an *xRQL* operation. We define *xRQL* operations for *creation*, *search*, *modification* and *deletion* of RDF metadata. RDF resource could be queried and modified with these four operations, while only *search* operation is provided for RDF schema.

3.2.2 RDF data description

RDF data description includes object element, objectvalue element, location element, condition element and namespace element.

- ***Object Element and ObjectValue Element***
  According to the consideration in section 3.1.2, we defined two elements to describe RDF resource under the object-oriented concepts. Object element is used to declare RDF resource class based on RDF schema. ObjectValue Element is used to assign unique ID and property values to RDF resource which belongs to the RDF resource class declared in object element.
- ***Location Element***
  A location element is defined to represent path expression, in which a sequence of classes and properties of RDF schema are used to describe RDF graph data model, described in section 3.1.2.
- ***Condition Element***
  A condition element is used to specify conditions for *search*, *modification* and *deletion* operations to RDF resource designated by the path expression in a location element.
- ***Namespace Element***
  A namespace element is defined to support multiple RDF schemas. The abbreviation of a namespace can be defined by a namespace element. For cases where the same class or property names are used in different schemas, the namespace element can be used to explicitly resolve such naming conflicts.

3.2.3 Result Format Description

In the *search* operation, a result format description element is provided to define the result format. It is based on XML syntax so that users can get a RDF query result in their favorite XML format for further processing.

3.3 RDF Schema Operations

Besides the operations for RDF resource description mentioned above, *xRQL* also supports RDF schema search operations.

3.3.1 Schema Variable and Path Expression

To distinguish the schema variables in the path expression, we proposed schema variables prefixed by "$". A schema path expression in which schema variables are depicted independently is expressed as:

<location>$x {managed} $y </location>

According to the layered data model mentioned in section 3.1.1, the *RDF Schema* level could be viewed as a instance of the *RDF Data Model* level. Hence, this schema path expression could be interpreted that $x belonging to *rdfs:Class* should be *domain* of *managed* property, while $y belonging to *rdfs:Class* should be *range* of *managed* property.

3.3.2 Function Definition

xRQL offers the functions of retrieving relationships of classes and properties from RDF schema, such as subclassof(), subpropertyof(), domain(), range(), typeof(), etc.

3.4 Examples of xRQL Operations

In this section, we depict the details of xRQL operations using typical examples.

Figure 5 shows a search operation for *Retrieving persons whose works are managed in companies*

```
<xrql:xrql>
<xrql:operator>select</ xrql:operator>
 <xrql:location>
     person:x{create}content{manage}company
 </xrql:location>
< xrql:result>
   <description>x</description>
</xrql:result>
<xrql:usingnamespace>
  <xrql:item>
     "ns1=www.docomo.co.jp/DE/metaschema"
</xrql:item>
</xrql:usingnamespace >
</xrql:xrql>
```

Figure 5: Example of search operation

In the location element, we use a data path expression with three class names (i.e. *person, content, company*) and two property names (i.e. *create, managed*). The data variable x represents the instances of the class *person*. Condition Element isn't used in this example. The Result Element designates the output format. Figure 6 shows the result of this *search* operation.

```
<xrql:result>

<description>"www.person1.co.jp"</description>
  <description>"www.person2.co.jp"</description>
  <description>"www.person3.co.jp"</description>
</xrql:result>
```

Figure 6: Result of Figure 5 search operation

By using several path expressions and condition requirements, more complicated queries for navigating through RDF repository are possible. Figure 7 shows a *search* operation to *Retrieve person and their names and telephone number whose first name is "John Smith"*.

In this query, we use two data path expressions. Three variables are defined in these two path expressions. Because either path expression starts from same class *person*, the path expressions implicit that both properties belong to the same class *person*. A Condition Element is defined in this example to specify a condition for data matching. The result format is also defined to retrieve three variables using different tags.

```
<xrql:xrql>
<xrql:operator>select</ xrql:operator>
<xrql:location>
       person:x{name}STRING:y,
        person{tel}STRING:z
</xrql:location>
 <xrql:condition>
 <xrql:item>y="john smith" </xrql:item>
 </xrql:condition >
< xrql:result>
     <resourcedescription>x</resourcedescription>
     <name>y</name>
     <tel>z</tel>
  </xrql:result>
  <xrql:usingnamespace>
     <xrql:item>
          "ns1=www.docomo.co.jp/DE/metaschema"
    </xrql:item>
  </xrql:usingnamespace >
</xrql:xrql>
```

Figure 7: Example of complex search operation

4 IMPLEMENTATION STATUS

We have implemented the xRQL processor over a native RDF management system. Unlike other RDF management implementations which usually manage RDF data by using relational databases, this system manages RDF metadata in a text file format. Figure 8 shows the system overview. The native RDF management system consists of an RDF parser, public API, Graph Match Process, Result Output Process and the Persistent Storage. We implement xRQL system by using the public APIs.

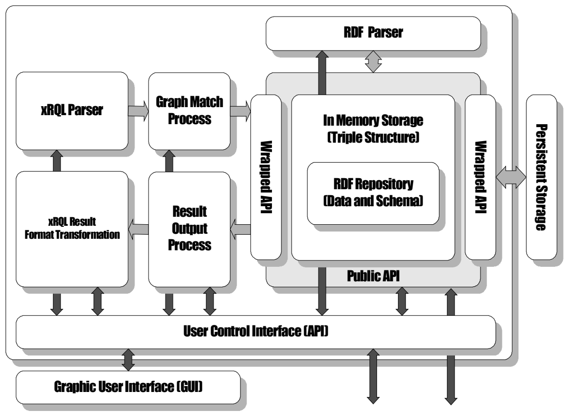
RDF Parser
xRQL Parser
Graph Match Process
In Memory Storage (Triple Structure)
RDF Repository (Data and Schema)
Wrapped API
Wrapped API
Persistent Storage
xRQL Result Format Transformation
Result Output Process
Public API
User Control Interface (API)
Graphic User Interface (GUI)

Figure 8: Overview of the native RDF Management System

4.1 User Control Interface

User Control Interface provides a friendly GUI for the users with convenient tools, such as query creation assistant and data input and output function. *xRQL parser* module and *xRQL Result Format Transformation* module work with other modules shown in Figure 8 to evaluate xRQL query, controlled by the User Control Interface.

4.2 xRQL Parser

xRQL parser analyzes an *xRQL* query string coming from *User Control Interface* and generates a query object. This query object will be delivered to *Graph Match Process* module to be evaluated. This parser is composed of the following three components.

- ***Preprocessor***
  The preprocessor component converts the input query string into standard format that could be processed by lexical and syntax analyzer.
- ***Lexical Analyzer***
  A simple XML parser (SAX) (SAX, 2002) is employed to perform the main parts of lexical analysis. Because some expressions defined in *xRQL* BNF are not completely encapsulated with XML tags, a small parser developed by JavaCC is used to parse these data.
- ***Syntax Analyzer***
  The syntax analyzer will extract tokens from Token Queue one by one. A corresponding query object will be generated once a syntax unit is matched. Finally, all query objects corresponding to sub syntax unit will be composed into a query object which represents the whole query.

4.3 xRQL Result Format Transformation

The result format information is analyzed by syntax analyzer. A result format object is created to preserve the information; the object is also integrated into the final query object. This transformation module extracts the result format object from the query object and uses it to render the query result received from *Graph Match Process*.

5 CONCLUSION AND FUTURE WORKS

In this paper, we propose an XML-based RDF query language to manipulate RDF schema and resource. Benefiting from the XML syntax, *xRQL* provides a machine-understand interface to the users in the XML-based distributed environment. Based on object-oriented concepts, manipulations of *search*, *deletion*, *creation* and *modification* of RDF resource are more pithy and effective. The result format definition also makes the result processing more flexible. *xRQL* has been implemented over a native RDF management system. Other functions, such as *Aggregation* and *Sort* functions will be considered in the future. Furthermore, the support of ontology inference (e.g. DAML+OIL) in *xRQL* is also an interesting topic in the future.

REFERENCES

Ora Lassila and Ralph R. Swick, 1999. "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation.

Dan Brickley and R.V. Guha, 2002. " RDF Vocabulary Description Language 1.0: RDF Schema", W3C Working Draft.

G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, Michel Scholl. 2002. “RQL: A Declarative Query Language for RDF”, WWW2002, Honolulu, Hawaii, USA. ACM 1-58113-449-5/02/0005.

Mike Olson, Uche Ogbuji. 2002. “RDF Query and Inference: Versa”, Available at http://rdfinference.org/versa.doc?xslt=db.xslt

L. Miller. 2001 “RDF Query using SquishQL”, Available at http://swordfish.rdfweb.org/rdfquery/.

Lee Jonas, Stefan Kokkelink. “RDF Path Requirements”, Available at: http://zoe.mathematik.uni-osnabrueck.de/RDFPath/req/

ODMG. 1998. “Object Query Language (OQL). User Manual”, Release 5.0.

D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. 2001. “XQuery: A Query Language for XML”, Working draft, World Wide Web Consortium. Available at http://www.w3.org/TR/xquery/.

S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. 1997. “The Lorel Query Language for Semistructured Data”, International Journal on Digital Libraries, 1(1):68{88.

Jeen Broekstra, Arjohn Kampman. (Sesame), 1999. “Query Language Definition”, EU-IST Project IST-1999-10132 On-To-Knowledge.

Serge Abiteboul. 1989. “Towards a deductive object-oriented database language”, the First International Conference on Deductive and Object-Oriented Databases,pages 453-472. North Holland, Amsterdam.

Simon Brown. 1999. “The semantic of object-oriented databases”, The University of Sheffield
Department of Computer Science.

Daniel Kim Chung Chan. 1993. “Object-oriented query language design and processing”, Doctor thesis. UNIVERSITY OF WISCONSIN-MADISON.

Zeki O. Bayram, Barrett R. Bryant, Faik Hakan Bilgen. 1996. “A Deductive Declarative Object-Oriented Data Model and Query Language based on Narrowing”, Proceedings of ISCIS XI, the 11th International Symposium on Computer and Information Sciences, Antalya, Turkey.

Flavius Frasincar Geert-Jan Houben Richard Vdovjak Peter Barna. 2002. “RAL: an Algebra for Querying RDF”, The Third International Conference on Web Information Systems Engineering(WISE'00) , p. 173. Singapore.

Joachim Peer. 2002. “A Logic Programming Approach to RDF Document And Query Transformation”, Workshop on Knowledge Transformation for the Semantic Web at the 15th European Conference on Artificial Intelligence. Lyon, France.

Nitish Manocha, Diane J. Cook, and Lawrence B. Holder. 2002. “Structural Web Search Using a Graph-Based Discovery System”, FLAIRS Conference: P133-137.

Jeonghee Kim, Taisook Han Kyu, Young Whang. 1999. “Visualization of Path Expressions in a Visual Object-Oriented Database Query Language”, 6th International Conference on Database Systems for Advanced Applications (DASFAA '99) Hsinchu, Taiwan, p. 99.

Martin Erwig, Ralf Hartmut Güting. 1994. “Explicit Graphs in a Functional Model for Spatial Databases”, IEEE Transactions on Knowledge and Data Engineering, Volume 6. Number 5, P.787-804.

R. Fikes. 2001. “DAML+OIL query language proposal”. Available at http://www.daml.org/listarchive/joint-committee/0572.html.

Edutella, 2002. “Edutella RDF Query Exchange Language”, Available at: http://www.kbs.uni-hannover.de/Diverses/edutella-archive/discussion/pdf00004.pdf.

T. Bray, J. Paoli, and C.M. Sperberg-McQueen. 1998. “Extensible markup language (XML) 1.0”,W3C Recommendation. Available at http://www.w3.org/TR/REC-xml/.

L. Sheng, Z. M. Özsoyoglu, G. Özsoyoglu. 1999. “A Graph Query Language and its Query processing”, IEEE ICDE Conf., Australia.

SAX, 2002 Available at: http://www.saxproject.org/