Preface

This volume contains the lecture notes of the Summer School “Reasoning Web 2006” (http://reasoningweb.org), which took place on September 4-6, 2006 in Lisbon and was hosted by the New University of Lisbon (Universidade Nova de Lisboa).

Like the first “Reasoning Web” Summer School (cf. LNCS 3564), which took place in 2005, the Summer School “Reasoning Web 2006” was organized by the Network of Excellence REWERSE, “Reasoning on the Web with Rules and Semantics” (http://rewerse.net), its member “Centre of Artificial Intelligence (CENTRIA)” at the New University of Lisbon being responsible for the local organization.

Reasoning is one of the central issues in Semantic Web research and development. Indeed, the Semantic Web aims at enhancing today’s Web with semantics-carrying “meta-data” and reasoning methods. The Semantic Web is a very active field of research and development, which involves both academia and industry.

The “Reasoning Web” Summer Schools provide a yearly forum for presenting and discussing recent developments in the “Semantic Web” field. They have a special focus on applied reasoning and on applications. They are primarily, but not only, intended for young researchers, especially PhD students and young professionals involved in research and/or development in the “Semantic Web” field.

The programme of the Summer School “Reasoning Web 2006” cover the following issues:

– Semantic Web Query Languages
– Semantic Web Rules and Ontologies
– Bioinformatics and Medical Ontologies
– Industrial Aspects

Semantic Web Query Languages. Query languages are expected to become as important on the Web and on the Semantic Web as they already are in databases. Indeed, many practical applications on today’s Web, and many of the Semantic Web applications that are expected to emerge, can be seen as information systems. Query languages ease the retrieval of data from complex databases or information systems. Query languages for the Web and the Semantic Web are an active area of research: in April 2006 the query language SPARQL, a query language for the Resource Description Framework RDF, attained the status of a “W3C Candidate Recommendation” (cf. http://www.w3.org/TR/rdf-sparql-query/); since 2004 a plethora of approaches to querying RDF have been proposed. The Summer School “Reasoning Web 2006” paid a tribute to this by including in its programme firstly a presentation of SPARQL by Bijan Parsia, a member of the “W3C RDF Data Access Working Group” which develops SPARQL, and secondly a comparative overview by Tim Furche,
Benedikt Linse, Dimitris Plexousakis, Georg Gottlob, and myself of selected query languages for RDF. This overview deepens and completes a first comparison presented at the Summer School “ReasoningWeb 2005”, which considered almost all query languages proposed for RDF but in a more superficial manner.

**Semantic Web Rules and Ontologies.** Rule-based formalisms currently receive considerable attention from Semantic Web researchers and developers: The W3C, for example, launched in November 2005 a “Rule Interchange Format (RIF)” Working Group (cf. [http://www.w3.org/2005/rules/](http://www.w3.org/2005/rules/)) and many researchers are now investigating how rule-based reasoning can be applied with XML, RDF, and/or OWL data. The Summer School “Reasoning Web 2006” therefore offered four complementary lectures on the subject. Two of them, given by Riccardo Rosati and by Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, and Hans Tompits, respectively presented recent approaches to rule-based reasoning with ontologies. A further lecture by Silvie Spreeuwenberg and Rik Gerrits was devoted to discussing the commonalities and the differences of “Business Rules” and “Semantic Web Rules”. A fourth and last lecture on rule-based formalisms for the Semantic Web by Uwe Aßmann, Jendrik Johannes, Jakob Henriksson, and Ilie Savga showed how modern software composition methods can be applied to Semantic Web rule languages.

**Bioinformatics and Medical Ontologies.** Bioinformatics and Medicine are a premier application field of Semantic Web methods. For this reason, Semantic Web researchers and developers can learn much from Semantic Web applications in these fields. The Summer School “Reasoning Web 2006” therefore offered three complementary lectures on Bioinformatics and Medical Ontologies: A first lecture by Alan Rector and Jeremy Rogers introduced the representation of medical concepts in the GALEN ontology; a second lecture by Michael Schroeder and Patrick Lambrix described a basis for a “Semantic Web for the Life Sciences”, and a third lecture by Ludwig Krippahl was devoted to the integration of Web data in the prediction of the structures and functions of proteins.

**Industrial Aspects.** Finally, the Summer School “Reasoning Web 2006” offered a lecture by Alain Léger, Johannes Heinecke, Lyndon J.B. Nixon, Pavel Shvaiko, Jean Charlet, Paola Hobson, and François Goasdoué on an industrial perspective of the Semantic Web.

Many persons contributed towards making the Summer School “Reasoning Web 2006” possible: First and foremost, the above mentioned lecturers; second the local organizers, in particular Carlos Viegas Damásio from the New University of Lisbon; and finally the programme committee consisting of Pedro Barahona, New University of Lisbon, Enrico Franconi, Free University of Bozen-Bolzano, Nicola Henze, University of Hannover, and Ulrike Sattler, University of Manchester, who all helped me in selecting the Summer School lectures and assessing their quality. Ulrike Sattler deserves a special mention for having collected the lecture notes and prepared this book. I would also like to mention Jan...
Małuszyński from the University of Linköping, and Norbert Eisinger from the University of Munich, coordinator and deputy coordinator of the REWERSE Working Group “Education and Training” on behalf of which the “Reasoning Web” Summer Schools are run.

I thank all of them warmly for their work, their dedication, and also for their lasting patience, which, I am afraid, was tried again and again during the eight months leading up to the summer school.

June 2006

François Bry
Organization

Programme Committee

Pedro Barahona  New University of Lisbon (Portugal)
François Bry  University of Munich (Germany), chair
Enrico Franconi  Free University of Bozen-Bolzano (Italy)
Nicola Henze  University of Hannover (Germany)
Ulrike Sattler  University of Manchester (UK)

Local Organisation

Carlos Viegas Damásio  New University of Lisbon (Portugal)

Sponsoring Institutions

The Reasoning Web 2006 Summer School is supported by the following organisations:

Fundação para a Ciência e a Tecnologia, Ministério da Ciência e Ensino Superior, Portugal.

Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Portugal.

<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Web Query Languages</strong></td>
</tr>
<tr>
<td>RDF Querying: Language Constructs and Evaluation</td>
</tr>
<tr>
<td>Methods Compared</td>
</tr>
<tr>
<td>Querying the Web with SPARQL</td>
</tr>
<tr>
<td><strong>Semantic Web Rules and Ontologies</strong></td>
</tr>
<tr>
<td>Composition of Rule Sets and Ontologies</td>
</tr>
<tr>
<td>Uwe Aßmann, Jendrik Johannes, Jakob Henriksson, Ilie Savga</td>
</tr>
<tr>
<td>Reasoning with Rules and Ontologies</td>
</tr>
<tr>
<td>Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, Hans Tompits</td>
</tr>
<tr>
<td>Integrating Ontologies and Rules: Semantic and Computational Issues</td>
</tr>
<tr>
<td>Riccardo Rosati</td>
</tr>
<tr>
<td>Business Rules in the Semantic Web, Are There Any or Are They Different?</td>
</tr>
<tr>
<td>Silvie Spreeuwenberg, Rik Gerrits</td>
</tr>
<tr>
<td><strong>Bioinformatics and Medical Ontologies</strong></td>
</tr>
<tr>
<td>Ontologies and Text Mining as a Basis for a Semantic Web for the Life Sciences</td>
</tr>
<tr>
<td>Andreas Doms, Vaida Jakoniené, Patrick Lambrix, Michael Schroeder, Thomas Wächter</td>
</tr>
<tr>
<td>Integrating Web Resources to Model Protein Structure and Function</td>
</tr>
<tr>
<td>Ludwig Krippahl</td>
</tr>
<tr>
<td>Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN</td>
</tr>
<tr>
<td>Alan Rector, Jeremy Rogers</td>
</tr>
</tbody>
</table>
Industrial Aspects

The Semantic Web from an Industry Perspective .......................... 232
   Alain Léger, Johannes Heinecke, Lyndon J.B. Nixon, Pavel Shvaiko,
   Jean Charlet, Paola Hobson, François Goasdoué

Author Index ................................................................. 269
RDF Querying: Language Constructs and Evaluation Methods Compared

Tim Furche¹, Benedikt Linse¹, François Bry¹, Dimitris Plexousakis²,³, and Georg Gottlob⁴

¹ Institute for Informatics, University of Munich, Oettingenstraße 67, 80538 München, Germany
http://www.pms.ifi.lmu.de/

² Department of Computer Science, University of Crete
Vassilika Vouton, P.O. Box 1385, GR 711 10 Heraklion, Crete, Greece

³ Information Systems Laboratory, Institute of Computer Science, FORTH
Vassilika Vouton, P.O. Box 1385, GR 711 10 Heraklion, Crete, Greece

⁴ Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
http://web.comlab.ox.ac.uk/oucl/people/georg.gottlob.html

Abstract. This article is firstly an introduction into query languages for the Semantic Web, secondly an in-depth comparison of the languages introduced. Only RDF query languages are considered because, as of the writing of this paper, query languages for other Semantic Web data modeling formalisms, especially OWL, are still an open research issue, and only a very small number of, furthermore incomplete, proposals for querying Semantic Web data modeled after other formalisms than RDF exist. The limitation to a few RDF query languages is motivated both by the objective of an in-depth comparison of the languages addressed and by space limitations. During the three years before the writing of this article, more than three dozen proposals for RDF query languages have been published! Not only such a large number, but also the often immature nature of the proposals makes the focus on few, but representative languages a necessary condition for a non-trivial comparison.

For this article, the following RDF query languages have been, admittedly subjectively, selected: Firstly, the “relational” or “pattern-based” query languages SPARQL, RQL, TRIPLE, and Xcerpt; secondly the reactive rule query language Algae; thirdly and last the “navigational access” query language Versa. Although subjective, this choice is arguably a good coverage of the diverse language paradigms considered for querying RDF data. It is the authors’ hope and expectation, that this comparison will motivate and trigger further similar studies, thus completing the present article and overcoming its limitation.
1 Introduction

Query Answering on the Semantic Web

Query answering is as central to the Semantic Web as it is to the conventional Web. Indeed, the Web as well as the emerging Semantic Web can be seen as information systems; and query answering is an essential functionality of any information system.

The Semantic Web is a research and development endeavor aiming at overcoming limitations of today’s Web. It has has been described as follows by W3C founder Tim Berners-Lee, Jim Hendler, and Ora Lassila:

“The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users.” [16]

In the Semantic Web, conventional Web data (usually represented in (X)HTML or other XML formats) is enriched by meta-data (represented, e.g., in RDF, Topic Maps, OWL) specifying the “meaning” of other data and allowing Web-based systems to take advantage of “intelligent” reasoning capabilities.

Query answering on the Semantic Web might be seen as more complex than querying on the conventional Web because the “meaning” conveyed by meta-data has to be properly “understood” and processed. In particular, query languages for RDF may convey RDF/S’s semantics as expressed, e.g., by RDF type triples.

Focus of this Article

This article is

1. an introduction into query languages for the Semantic Web;
2. an in-depth comparison of the languages introduced along prominent language constructs and concepts.

Only RDF query languages are considered in this article. The reason for this is, that as of the writing of this paper, query languages for other Semantic Web data modeling formalisms, especially OWL, still are an open research issue, and only a very small number of, furthermore incomplete, proposals for querying Semantic Web data modeled after other formalisms than RDF are known.

Furthermore, only a few RDF query languages are considered in this article. This limitation is motivated both by the objective of an in-depth comparison of the languages addressed and by space limitations. During the three years before the writing of this article, more than three dozen proposals for RDF query languages have been published! Not only such a large number, but also the often immature nature of the proposals makes the focus on few, but representative languages a necessary condition for a non-trivial comparison.

In the spirit of a practical introduction into these query languages, we have taken an example-centered approach. We believe that this is advantageous to
the reader to quickly gain an impression of the language and constructs. Furthermore, a more formal treatment of the languages is impeded by the lack of (published) formal semantics. In Section 5, however, different semantics for interesting language constructs are addressed and compared in select cases.

This article builds upon and complements the survey of Semantic Web query languages co-authored in 2005 by some of the authors of the present article.

While the focus of the 2005 survey has been a complete, but therefore necessarily somewhat shallow coverage of Semantic Web query languages, including on the one hand query languages for Topic Maps and on the other hand all known “dialectal” variations of RDF query languages. In contrast, the present article is focused on an in-depth comparison of a few selected RDF query languages that the authors consider representative. Although building upon the survey, this article is self-contained.

At least the first part, of the article is mostly of an introductory nature. We believe, however, that also researchers and scientists already acquainted with RDF query languages can benefit from the presented material. This applies particularly to the comparison of language constructs and evaluation methods in the second part. We hope that the direct comparisons reveal choices that language designers face when deciding which constructs to support in which way, and that language users face when deciding which languages are suitable for their particular needs.

Language Selection and Order

This article aims at introducing from the perspective of the authors interesting and representative selection of query languages proposed for RDF:

– Firstly, the “relational” or “pattern-based” query languages SPARQL, RQL, TRIPLE, and Xcerpt (with its visual “twin” visXcerpt).

– Secondly, the “reactive rule” query language Algae.

– Thirdly, the “navigational access” query language Versa.

Although incomplete and admittedly subjective, this choice can be seen as a good coverage of the diverse language paradigms considered for querying RDF data.

It is the authors’ hope and expectation that this comparison will motivate further similar studies that complete the present article and overcome its limitation. It is also the authors’ hope that this article will provide Semantic Web practitioners and researchers alike with a good introduction into query answering on the Semantic Web even though it does not address all query languages proposed for the Semantic Web.

Structure of this Article

The following three questions are at the heart of this article and give it its structure:

1 Sections 2 and 3 are shortened versions of corresponding sections of 5.
1. what are the core paradigms of each query language,
2. what language constructs do different languages offer to solve tasks such as path traversal, optional selection, or grouping,
3. how are they realized?

In Section 2 the RDF/S data model, a running example, the RDF/S semantics and serialization formats are introduced. Section 3 begins by presenting a categorization of Semantic Web queries and sample queries for each category. Subsequently, in Section 4 the RDF query languages selected are introduced—grouped according to their families, i.e., “relational” or “pattern-based”, “reactive rule” and “navigational access”. For each language considered, some of the sample queries are formulated. For the sake of conciseness and simplicity, not all sample queries are expressed in each language considered. In Section 5 a summary and comparison of language features observable and desirable for RDF query languages is given. Section 6 examines evaluation methods of Semantic Web queries. Section 7 concludes this survey.

2 A Brief Introduction to RDF and RDFS

2.1 Data Model

RDF [10, 59] data are sets of “triples” or “statements” of the form (Subject, Property, Object). RDF data are commonly seen as directed graphs the nodes of which are statement’s subjects and objects and the arcs of which correspond to statement’s properties, i.e., an arc relates a statement’s subject with the statement’s object. Properties are also called “predicates”. Nodes (i.e., subjects and objects) are either

1. labeled by URIs describing Web resources,
2. or labeled by literals, i.e., scalar data such as strings or numbers,
3. or are unlabeled and called anonymous or “blank nodes”.

Blank nodes are commonly used to group or “aggregate” properties. Specific properties are predefined in the RDF and RDFS recommendations [21, 53, 59, 69], e.g., rdf:type for specifying the type of resources, rdfs:subClassOf for specifying class-subclass relationships between subjects/objects, and rdfs:subPropertyOf for specifying property-subproperty relationships between properties. Furthermore, RDFS has “meta-classes”, e.g., rdfs:Class, the class of all classes, and rdf:Property, the class of all properties.

RDFS [21] allows one to define so-called “RDF Schemas” or “ontologies”, similar to object-oriented data models. The inheritance model of RDFS exhibits the following peculiarities:

1. resources can be classified in different classes that are not related in the class hierarchy,
2. the class hierarchy can be cyclic so that all classes on the cycle are “subclass equivalent”,
3. properties are first-class objects, and
4. RDF does not describe which properties can be associated with a class, but
   instead the domain and range of a property.

Based on an RDFS schema, “inference rules” can be specified, for instance the
transitivity of the class hierarchy, or the type of an untyped resource that has a
property associated with a known domain.

RDF can be serialized in various formats, the most frequently used being
(RDF/) XML. Early approaches to RDF serialization have raised considerable
criticism due to their complexity. As a consequence, a surprisingly large number
of RDF serializations have been proposed, cf. [26] for a detailed survey.

2.2 Running Example: Classification-Based Book Recommender

In the following, queries in a simple book recommender system describing vari-
ous properties and relationships between books are considered as running ex-
amples. The recommender system describes properties of and relationships
between books. It consists of a hierarchy (or ontology) of the book categories
Writing, Novel, Essay, Historical_Novel, and Historical_Essay, and two books The
First Man in Rome (a Historical_Novel authored by Colleen McCullough) and
Bellum Civile (a Historical_Essay authored by Julius Caesar and Aulus Hirtius,
and translated by J.M. Carter). Figure 1 depicts these data as a (simplified)
RDF graph [21, 59, 63]. Note in particular that a Historical_Novel is both, a
Novel and an Essay, and that books may optionally have translators, as is the
case for Bellum Civile.

The simple ontology in the book recommender system only makes use of the
subsumption (or “is-a-kind-of”) relation rdfs:subClassOf and the instance (or “is-
a”) relation rdf:type. This simple and small ontology is sufficient to illustrate the
most important aspects of RDF query languages.

The RDF representation of the sample data refers to the “simple datatypes” of
XML Schema [17] for scalar data: Book titles and authors’ names are “strings”,
(untyped or typed as xsd:string), publication years of books are “Gregorian
years”, xsd:gYear. The sample data are assumed to be accessible at the URI
http://example.org/books#. Where useful, e.g, when referencing the vocabu-
lary defined in the ontology part of the data, this URL is associated with the
prefix books.

Representation of the Sample Data in RDF. The RDF representation of
the book recommender system directly corresponds to the simplified RDF graph
in Fig. 1. It is given here in the Turtle serialization.

```turtle
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

3 The same example is used in [5].
Fig. 1. Sample Data: representation as a (simplified) RDF graph

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.org/foaf/0.1/> .
:Writing a rdfs:Class ; rdfs:label "Novel" .
:Novel a rdfs:Class ; rdfs:label "Novel" ;
  rdfs:subClassOf :Writing .
:Essay a rdfs:Class ; rdfs:label "Essay" ;
  rdfs:subClassOf :Writing .
:Historical_Essay a rdfs:Class ;
  rdfs:label "Historical Essay" ;
  rdfs:subClassOf :Essay .
:Historical_Novel a rdfs:Class ;
  rdfs:label "Historical Novel" ;
  rdfs:subClassOf :Essay ;
  rdfs:subClassOf :Novel .
:author a rdf:Property ;
  rdfs:domain :Writing ;
  rdfs:range foaf:Person .
:translator a rdf:Property ;
  rdfs:domain :Writing ;
  rdfs:range foaf:Person .
_:b1 a :Historical_Novel ;
  :title "The First Man in Rome" ;
  :year "1990"^^xsd:gYear ;
  :author [foaf:name "Colleen McCullough"] .
_:b2 a :Historical_Essay ;
  :title "Bellum Civile" ;
  :author [foaf:name "Julius Caesar"] ;
  :author [foaf:name "Aulus Hirtius"] ;
  :translator [foaf:name "J. M. Carter"] .
Books, authors, and translators are represented by blank nodes without identifiers, or with temporary identifiers indicated by the prefix “_:”.

2.3 Semantics

The meaning of RDF data (e.g., what means “book”? ) cannot be fully understood by applications and is interpreted in different ways also by human readers. Naturally, it depends on social, cultural, temporal and other types of context information. However, RDF/S allow to specify part of the semantics of applications (e.g., “a book might have an author”).

As is common practice for declarative languages, the semantics of RDF/S is specified in terms of a model theory \[39, 53\]. RDF applications should be able to derive information using the inference rules for basic RDF, while only schema-aware applications are expected to take into account information provided by RDFS inference rules.

3 Sample Queries

The RDF query languages considered in this article are illustrated and illustrated using five different types of queries against the sample data. This categorization is inspired by \[67\] and \[34\].

Selection queries simply retrieve parts of the data based on its content, structure, or position. The first query is thus:

**Query 1.** “Select all Essays together with their authors (i.e. author items and corresponding names)”

Extraction queries extract substructures, and can be considered as a special form of Selection Queries returning not only explicitly queried resources or statements, but entire subgraphs.

**Query 2.** “Select all data items with any relation to the book titled ‘Bellum Civile’.”

Reduction queries: Some queries are more concisely expressed by specifying what parts of the data not to include in the answer:

**Query 3.** “Select all data items except ontology information and translators from the book recommender system.”

Restructuring queries: In Web applications, it is often desirable to restructure data, possibly into different formats or serializations. For example, the contents of the book recommender system could be restructured to an (X)HTML representation for viewing in a browser, or derived data could be created, like inverting the relation author:

\[ Again, these queries are mostly the same as in \[5\].\]
Query 4. “Invert the relation author (from a book to an author) into a relation authored (from an author to a book).”

In particular, RDF requires restructuring for reification, i.e. expressing “statements about statements”. When reifying, a statement is replaced by four new statements specifying the subject, predicate, and object of the old statement. For example, the statement “Julius Caesar is author of Bellum Civile” is reified by the four statements “X is a statement”, “X has subject Julius Caesar”, “X has predicate author”, and “X has object Bellum Civile”.

Aggregation queries: Restructuring the data also includes aggregating several data items into one new data item. As Web data usually consists of tree- or graph-structured data that goes beyond flat relations, we distinguish between value aggregation working only on the values (like SQL’s \texttt{max(·)}, \texttt{sum(·)}, ...) and structural aggregation working also on structural elements (like “how many nodes”). Query\texttt{5} uses the \texttt{max(·)} value aggregation, while Query\texttt{6} uses structural aggregation:

Query 5. “Return the last year in which an author with name ‘Julius Caesar’ published something.”

Query 6. “Return each of the subclasses of ‘Writing’, together with the average number of authors per publication of that subclass.”

Combination and inference queries: It is often necessary to combine information that is not explicitly connected, like information from different sources or substructures. Such queries are useful with ontologies that often specify that names declared at different places are synonymous:


Combination queries are related to inference, because inference refers to combining data, as illustrated by the following example: If the books entitled ‘Bellum Civile’ and ‘The Civil War’ are the same book, and ‘if ‘Julius Caesar’ is an author of ‘Bellum Civile’, then ‘Julius Caesar’ is also an author of ‘The Civil War’. Inference queries e.g. compute transitive closures of relations like the RDFS subClassOf relation:

Query 8. “Return the transitive closure of the subClassOf relation.”

Not all inference queries are combination queries, as the following example illustrates:

Query 9. “Return the co-author relation between two persons that stand in author relationships with the same book.”

Some query languages have closure operators applicable to any relation, while other query languages have closure operators only for certain, predefined relations, e.g., the RDFS subClassOf relation. Some query languages support general recursion, making it possible and easy to express the transitive closure of every relation.
4 The RDF Query Language Families

In this survey, we focus on three groups of RDF query languages differing in what the authors perceive as central paradigms of the languages. Languages following the relational or pattern-based paradigm use selection constructs similar to selection-projection-join (SPJ) queries. Though they share a common query core, the languages in this group vary quite noticeably, some extending SPJ queries very conservatively, others going well beyond with novel constructs aiming to adequately support the specifics of RDF. The second group is set apart by the use of reactive rules but otherwise shares some commonality with the first group. The final group is more distinctly separated by preferring navigational access and path expressions over patterns.

Figure 2 may serve as orientation through the “language zoo” discussed in this chapter and includes also “dialects” and variants that are only briefly mentioned in the following.

4.1 The Relational Query Languages SPARQL, RQL, TRIPLE, and Xcerpt

The SPARQL Family. SPARQL [84] is a query language that has already reached candidate recommendation status at the W3C, and is on a good way to become the W3C recommendation for RDF querying. It has its roots in SquishQL [76] and RDQL [91].

Querying RDF data with languages in the SPARQL family amounts to matching graph patterns that are given as sets of triples of subjects, predicates and objects. These triples are usually connected to form graphs by means of joins expressed using several occurrences of the same variable. SPARQL uses the Turtle [7] serialization format for RDF as basis for its own triple syntax. It inherits certain syntactic shorthands from Turtle: e.g., predicate-object lists allow several statements to share the same subject without repeating the subject. Pairs of predicates and objects following the subject are separated by colons. Object lists are shorthands for several statements sharing both the subject and the predicate, the objects being separated by commas.

Solutions to SPARQL (or SquishQL or RDQL) queries are given in the form of result sets, for which also an XML format has been specified [9]. In SPARQL, result sets are sets of mappings from the variables occurring within the query to nodes of the queried data. Although RDQL and SquishQL are predecessors of SPARQL, this section presents realizations of the sample queries only in SPARQL. The formulation in the other members of the SPARQL family are very similar though some of the queries use features only recently added and not available in RDQL and SquishQL.

In SPARQL, Query 1 is expressed as follows.

```sparql
PREFIX books: <http://example.org/books#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

---

Fig. 2. Chronological Overview of RDF Query Languages (in bold typeface: languages covered in this survey; in italic typeface: non-RDF (mostly XML) query languages with proposals/extensions for querying RDF; MetaLog’s unique approach to RDF querying based on a natural language interface defies classification in this framework); N3QL is not classified due to incomplete description

SELECT ?essay ?author ?authorName
FROM <http://example.org/books>
  ?author books:name ?authorName . }

The WHERE clause specifies the graph pattern to match using variables to select data. Variables are recognized by either ? or $ prefix. Triples are connected to graph patterns using “.” (colon). The FROM clause specifies the URL (or some other identifier) of the data to be queried and the SELECT clause the result variables.

Extraction queries like Query 2 can only be approximately expressed in all members of the SPARQL family, because recursive traversals of the data are not possible. Thus one cannot extract all information relevant to a particular resource. Collecting all outgoing edges of a node together with the directly linked objects of these predicates is possible and is showcased in the sample code below.
As can be seen, SPARQL does not syntactically differentiate between variables for predicates and for resources, as opposed to RQL discussed below. Also the extraction of information occurring at a fixed distance from the resource representing the book named “Bellum Civile” is possible by adding further statements to the query below.

```
PREFIX books: <http://example.org/books#>
SELECT ?property ?propertyValue
FROM <http://example.org/books>
WHERE {?essay books:title "Bellum Civile" . ?essay ?property ?propertyValue . }
```

Another way to approximate extraction queries are SPARQL’s DESCRIBE queries that allow the retrieval of “descriptions” for resources. The exact extent of such a “description” is not defined in [84], but concise bounded descriptions [96] are referenced as a reasonable choice. These represent a form of predefined extraction query that returns all immediate properties for a resource as well as the immediate properties of all blank nodes that are reachable from the resource to be described without other named resources in between.

The FILTER keyword is used in SPARQL to eliminate result sets which evaluate to false when substituted in the boolean expressions given in the body of the FILTER clause. A query that finds the persons that have authored a book with title “Bellum Civile” can be expressed in SPARQL as follows:

```
PREFIX books: <http://example.org/books#>
SELECT ?person
FROM <http://example.org/books>
```

The three queries mentioned above are also expressible in SPARQL’s predecessors SquishQL and RDQL with a slightly different syntax but almost identical structure. SPARQL and its relatives do not support RDF/S inferencing, which means that among other tasks, querying all resources of type books:Writing of the example data above would not return any results, because there are no resources which are directly associated with books:Writing via an rdf:type property. If the SPARQL family provided support for inferencing, the resources represented by the blank nodes _:b1 and _:b2 in the serialization in Section 2.2 could be returned as results to the query in compliance with the rule RDFS9 of the RDFS semantics. One can argue that RDF/S and OWL reasoning should not be a task of the query language, but should be provided by an underlying black box reasoner. Given such a reasoner that transparently provides the full RDFS entailment graph, i.e., the closure graph under the RDF/S inference rules, the languages of the SPARQL family can very well answer queries such as the one just mentioned.

There are several other characteristics and also limitations of the members of the SPARQL family, which deserve to be mentioned: