Building a Semantic Web System for Scientific Applications: An Engineering Approach

Renato Fileto[1,3], Claudia Bauzer Medeiros[1], Calton Pu[2], Ling Liu[2], and Eduardo Delgado Assad[3]

[1] Institute of Computing, University of Campinas,
Caixa Postal 6176, Campinas, SP, 13081-970, Brazil
{fileto, cmbm}@ic.unicamp.br
[2] College of Computing, Georgia Institute of Technology,
801 Atlantic Drive, Atlanta, GA, 30332-0280, USA
{lingliu, calton}@cc.gatech.edu
[3] Embrapa – Brazilian Agricultural Research Agency,
Av. Dr. Andre Torsello, 209, Campinas, SP, 13083-886, Brazil
{fileto, assad}@cnptia.embrapa.br

Abstract. This paper presents an engineering experience for building a Semantic Web compliant system for a scientific application – agricultural zoning. First, we define the concept of ontological cover and a set of relationships between such covers. These definitions, based on domain ontologies, can be used, for example, to support the discovery of services on the Web. Second, we propose a semantic acyclic restriction on ontologies which enables the efficient comparison of ontological covers. Third, we present different engineering solutions to build ontology views satisfying the acyclic restriction in a prototype. Our experimental results unveil some limitations of the current Semantic Web technology to handle large data volumes, and show that the combination of such technology with traditional data management techniques is an effective way to achieve highly functional and scalable solutions.

1 Introduction

POESIA (Processes for Open-Ended Systems for Information Analysis) [4, 5] pursues the Semantic Web vision [1, 13] to bring about solutions for resources discovery and composition on the Web. The foundations of POESIA are: (1) Web services to encapsulate data sets and processes; (2) workflow technology to manage complex processes; and (3) domain ontologies to drive the description, discovery and composition of resources. POESIA's mechanisms for composing Web services appear in [5], and its methods for tracking data provenance and support data integration appear in [6].

This paper focuses on the design and implementation challenges of handling domain ontologies in POESIA. In particular, it points out the obstacles met in loading and using domain ontologies in application programs, and describes the solutions implemented in a prototype. Rather than forcing applications to deal with large, cumbersome ontologies, we propose ontology views satisfying the acyclic restriction to enable efficient automated means to discover and compose Web services based on domain specific knowledge. In these ontology extracts, one can determine the relative order of ontology terms by directed graph traversal or just one string comparison, instead of using inference engines, for example.

The experimental results of our implementation effort give an insight on the limitations of the current Semantic Web technology, when faced with applications using large data sets. The combination of Semantic Web standards and tools with conventional data management techniques provides more efficiency and scalability than the solutions based purely on Semantic Web technologies.

2 Motivating Application: Agricultural Zoning

This research has been inspired by the need of versatile tools to support scientific applications on the Web, and more specifically the development of decision support systems for agriculture. One example of an application in this domain is *agricultural zoning* – a scientific process that classifies the land in a given geographic region into parcels, according to their suitability for a particular crop, and the best time of the year for key cultivation tasks (such as planting, harvesting, pruning, etc). The goal of agricultural zoning is to determine the best choices for a productive and sustainable use of the land, while minimizing the risks of failure. It requires looking at many factors such as climate, regional topography, soil properties, crop requirements, social and environmental issues.

Typically, this kind of application involves intricate data processing activities across different organizations. Agricultural zoning relies on data from a variety of heterogeneous sources, including sensors that collect data on physical and biological phenomena (e.g., weather stations, satellites, and laboratory automation equipment). These data may be stored in a variety of databases, with different spatial, temporal and thematic scopes. Domain experts combine these data, in multiple steps of a multi-institucional process, in order to produce, for example, maps showing the suitability of the lands of a particular state for planting soybeans in different periods of the year.

3 Solution Context

POESIA relies on Web services [2] to encapsulate data sets and processes, so that Web standards and protocols ensure interoperability among different platforms. Ontologies, which can also be published and looked up through Web services, play another key role in POESIA. They provide a shared conceptualization to drive the description, discovery and composition of distributed resources.

3.1 General Architecture

Figure 1 illustrates the general architecture of a POESIA supporting system. In the right bottom corner, it shows the three kinds of POESIA servers connected

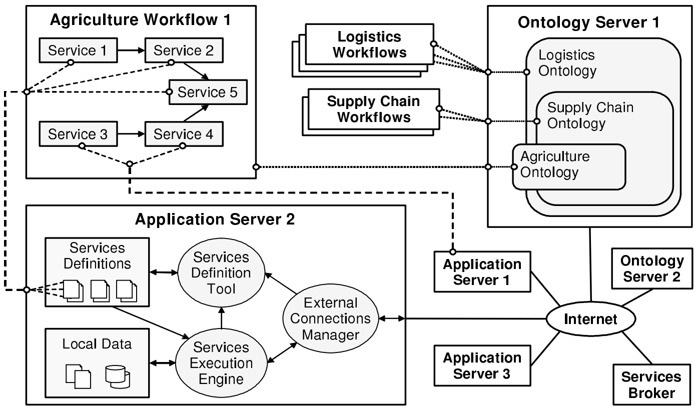


Fig. 1. POESIA's architecture

through the Internet: application servers, ontology servers, and service brokers. An *Application Server* maintains a local data set and a collection of service definitions that provide external access to local resources. Mechanisms for creating composite services are specified and managed by means of workflows, so that arbitrarily complex processes can be built from other processes [5].

An *Ontology Server* offers encapsulated access to a set of ontologies about different domains (e.g., agriculture, logistics), with adaptation means for particular application needs, including the extraction of ontology views. A *Services Broker* is a catalog of resources that centralizes the discovery of services using ontology views. The sharing of an ontology view among application servers and service brokers enables the semantic-driven discovery and composition of services.

3.2 An Ontology for Agriculture

As part of the effort to implement and validate POESIA, we have developed an ontology to support agricultural zoning. This ontology is divided in *dimensions* – ontology portions referring to particular agricultural concerns and disconnected from each other. Figure 2 illustrates the `Territory` dimension, depicting 3 layers of geographic data that refer to independent territorial partitions: political divisions, ecological regions and hydrological basins. These layers have `Country`, `Eco Region` and `Macro Basin` as their respective top classes.

Each rectangle in Figure 2 represents a *class*. Edges ending with a diamond represent *specialization relations* (of type IS_A) – the class at the diamond side is a subclass of the class in the other end of the edge. *Aggregation relations* (of type PART_OF) are represented by edges with a black circle on the side of the class playing the constituent role. Other dimensions of the agriculture ontology (e.g., `Agricultural Product`) are also represented with the basic constructs described above.

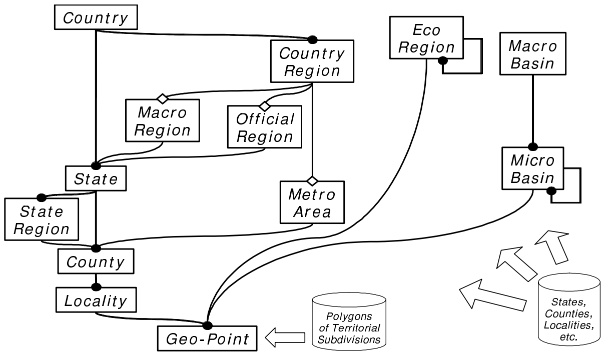


Fig. 2. The `Territory` dimension

4 Using the Domain Ontology

A POESIA ontology view has the form of a directed acyclic graph Σ, whose nodes refer to terms that can be concepts (e.g., `Country`) or instances of concepts (e.g., `Country(Brazil)`). The directed edges of Σ refer to semantic relations between terms (`instantiation`, `specialization` or `aggregation`). Edges are oriented from the general to the instantiated, specialized or constituent terms. These semantic relations induce a partial order among the terms [5], determined by the relative positions of these terms in the ontology graph. Imposing the acyclic restriction preserves the semantic contents of the original ontology, because it requires removing just inverse relations (e.g., remove generalizations while maintaining the corresponding specialization relations). Figure 3 shows a POESIA ontology view, where instances of some classes from the ontology dimension of Figure 2 appear in the right bottom corner.

4.1 The Encompass Relation, Ontological Covers and Service Scope

Let t and t' be two terms of an ontology view Σ. We say that t *encompasses* t', denoted by $t \models t'$, if and only if there is a path in Σ leading from t to t', i.e., a sequence of instantiations, specializations and/or aggregations relating t to t'.

The encompass relation is `transitive` – if Σ has a path from t to t' and another path from t' to t'', then Σ has a path from t to t''. In the right bottom corner (`Territory` dimension) of the view in Figure 3, for example, `Country(Brazil)` $\models$ `State(RJ)` and `State(RJ)` $\models$ `State(RJ).County(Valena)`. The string `State(RJ).County(Valena)` represents the path to the term `County(Valena)` in Rio de Janeiro State.

An *ontological cover* is a tuple of terms taken from different dimensions of an ontology view. For instance, the ontological cover `[Orange, Country(Brazil)]` is a tuple of terms from two dimensions of a POESIA view of the agricultural ontology – `Agricultural Product` and `Territory`.

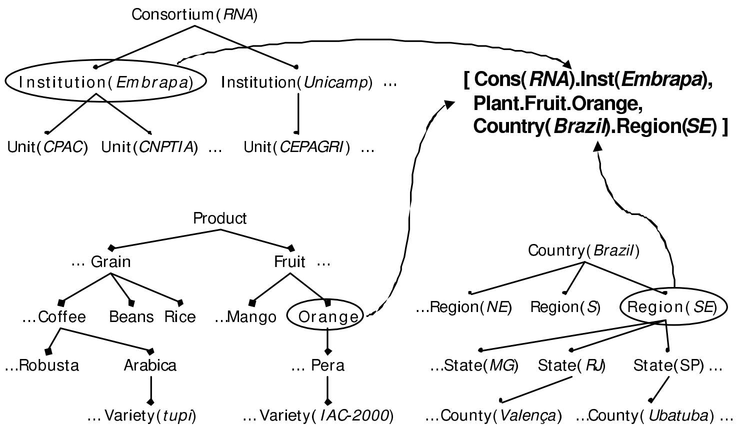

Consortium(RNA)
Institution(Embrapa)
Institution(Unicamp) ...
Unit(CPAC) Unit(CNPTIA) ... Unit(CEPAGRI) ...
[Cons(RNA).Inst(Embrapa), Plant.Fruit.Orange, Country(Brazil).Region(SE)]
Product
... Grain
Fruit ...
...Coffee Beans Rice ..Mango Orange
...Robusta Arabica ... Pera
... Variety(tupi) ... Variety(IAC-2000)
Country(Brazil)
...Region(NE) Region(S) Region(SE)
...State(MG) State(RJ) State(SP) ...
...County(Valença) ..County(Ubatuba) ...

Fig. 3. An ontological cover in the agriculture ontology

An ontological cover attached to a Web service plays the role of metadata, describing the utilization scope of that service. We thus define the *service scope* as the transitive closure of the nodes reachable from the terms of the ontological cover associated with a particular service. Hence, the ontological cover `[Orange, Country(Brazil)]`, when attached to a Web service providing access to agricultural production data, indicates that data from that service refer to the production of `Oranges` in `Brazil`. Figure 3 illustrates the composition of an ontological cover in which the term `Institution(Embrapa)` expresses the utilization scope in the `Organization` dimension, `Orange` in the `Product` dimension and `Country(Brazil).Region(SE)` in the `Territory` dimension.

4.2 Relations Between Ontological Covers and Services Discovery

The *encompass* relation between terms gives rise to corresponding relations between ontological covers. For simplicity, let us consider that an ontological cover has exactly one term for each dimension[1]. Given two ontological covers, $OC = [t_1, \cdots, t_n]$ and $OC' = [t'_1, \cdots, t'_n]$ $(n \geq 1)$, where $t_i \in OC$ and $t'_j \in OC'$ are terms from the same ontology view Σ, OC and OC' may be disjoint or satisfy one of the following relations.

Overlapping: *OC overlaps OC'* if and only if:

1. $\forall\, t \in OC : \exists\, t' \in OC'$ such that $t \models t' \;\vee\; t' \models t$
2. $\forall\, t' \in OC' : \exists\, t \in OC$ such that $t \models t' \;\vee\; t' \models t$

Encompassing: $OC \models OC'$ if and only if:

1. $\forall\, t \in OC : \exists\, t' \in OC'$ such that $t \models t'$
2. $\forall\, t' \in OC' : \exists\, t \in OC$ such that $t \models t'$

[1] Multiple terms referring to the same dimension are considered in [4, 5].

Equivalence: $OC \equiv OC'$ if and only if:

1. $\forall\, t \in OC : \exists\, t' \in OC'$ such that $t \models t'$
2. $\forall\, t' \in OC' : \exists\, t \in OC$ such that $t' \models t$

Overlap is bidirectional and the weakest of these relations. The *encompass* relation, on the other hand, only accepts encompassing relations between terms in one direction. The *equivalence* relation requires that each pair of terms taken from the two ontological covers reciprocally encompass each other. Finally, two ontological covers are *disjoint* if they do not overlap each other in at least one dimension, i.e., there is a term in one of the covers that does not encompass neither is encompassed by any term of the other cover.

The *encompass*, *overlap* and *equivalence* relations between ontological covers are *reflexive* and *transitive*, and the two latter are also *symmetric*. The transitiveness of these relations induces a partial order among ontological covers referring to the same ontology.

Services discovery can be understood as an ontology based query. More specifically, a service discovery request is stated as a query specified in the same way as an ontological cover, i.e., as a collection of terms from a POESIA ontology view. The query processing corresponds to investigating relations between the query and the service scopes, all of which are expressed by ontological covers. The services satisfying a query q are those whose ontological covers overlap q.

5 Engineering Considerations: Design and Implementation

Our ontology for agriculture has been built with Protg [9], an open-source graphic tool for ontology construction. POESIA's current implementation accepts ontologies in the RDF format [11] exported by Protg.

The acyclic restriction imposed on ontology views enables more efficient algorithms for comparing ontological covers than using, for example, inference engines to process a full ontology. Thus, our engineering solution to handle ontologies in POESIA involves three aspects: (i) use views tailored for particular applications, thereby reducing the number of terms and relations to be handled; (ii) restrict these ontology views to directed acyclic graphs (DAGs) or trees, in order to enable efficient algorithms to check relations between ontological covers; (iii) adopt a procedural approach to ontology management, backed by databases to attain persistence and scalability.

Our view extracting algorithm is analogous to that of [8], as it works by traversing the ontology graph. The view specification consists of three sets: starting classes, intermediate classes and properties (semantic relationships) to traverse. The algorithm traverses the class and instances hierarchy from the starting classes, including in the view only the classes, instances and properties contained in the view specification. The classes and properties of the view specification must ensure the acyclic restriction.

OntoCover, our prototype Java package for building ontology views, uses the Jena toolkit [15] to parse and handle RDF statements. The RDFS (RDF-Schema) file delineates the classes, subclasses and properties of an ontology, with tags marking the classes and properties of the view specification described above. The RDF file, on the other hand, contains the instances of those classes and their respective properties. Jena loads RDF/RDFS files in memory or in a database management system (DBMS) and allows navigation in the RDF triples through the Jena API or the RDQL query language [12]. The DBMS provides persistence and scalability for large ontology specifications.

We construct an ontology view by using Jena in two steps: (1) load in RAM the RDFS that specifies the ontology view; and (2) manipulate this RDFS specification via Jena API to generate the view, considering three alternatives for getting the many instances of the ontology classes to include in the view:

RAM: use Jena to parse RDF specifications from files into an auxiliary data structure in RAM, manipulated via the Jena API to build the view tree;
DB RDF: use the Jena API to handle ontology instance data stored in PostgreSQL [10] as RDF triples;
DB Conventional: take instances from a PostgreSQL database that contains one table per ontology class.

The database schema used by Jena to store RDF triples in the DBMS – for the `DB RDF` strategy – appears in [15]. The schema employed by the `DB Conventional` strategy to maintain the `Territory` dimension instances (for our experiments) is the one presented in Figure 2, regarded as an entity-relationship diagram, i.e., when each rectangle represents an entity and each edge represents a `1:N` relationship, with the diamond or black circle in the `N` side.

6 Performance Evaluation

6.1 Comparing Ontological Covers on Ontology Views

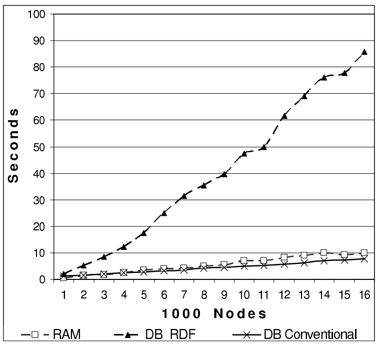
Using views with the acyclic restriction, one can efficiently determine semantic relations between ontology terms. In a tree-like ontology view, determining if a term t encompasses another term t' reduces to checking if the string representing the path from a root o of the view to t is the head of the string representing the path from o to t'. Therefore, the problem is solved by just one string comparison. In a DAG-like view, one can use graph search algorithms to determine if there is a path from t to t'. These algorithms run in linear time for the ontology views used in agricultural zoning applications, because these views have the number of edges (semantic relations) proportional to the number of nodes (terms).

6.2 View Construction

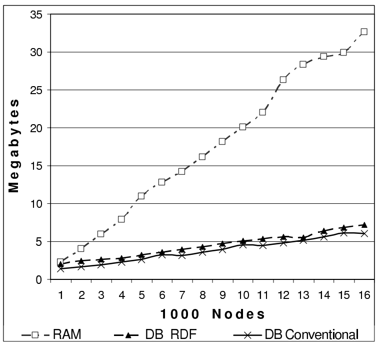
Given our engineering option for ontology views satisfying the acyclic restriction, the bottleneck has been the memory and time necessary for extracting the views. Therefore, we focused our experiments on this part of the solution, comparing the alternatives described in Section 5 for managing ontologies stored as RDF/RDFS files and relational databases.

Our experiments used the ontology described in Section 3.2. Instances for the `Territory` dimension of this ontology were provided by IBGE (Brazilian Institute of Geography and Statistics), yielding an ontology view graph with more than 15000 nodes, to allow experiments with large volumes of data. These experiments ran on Linux (Red Hat 8), in a 1.6 GHz Pentium IV machine, with 512 megabytes of RAM.

Figure 4 presents the results of some experiments on constructing tree-like views of the agriculture ontology, with chunks of increments of 1000 nodes, as shown in the X-axis. The Y-axis represents the time to build the view (Figure 4(a)) or the memory use (Figure 4(b)). We compare the strategies described in Section 5; namely, `RAM`, `DB RDF` and `DB Conventional`. For the `RAM` strategy, we consider the time to parse RDFS and RDF, plus the time to build the tree by handling these RDF specifications in memory. `DB RDF` and `DB Conventional`, on the other hand, rely on the efficiency of a DBMS to manage large data sets in persistent memory. These strategies only load RDFS as a whole in memory, and query individual instances of the ontology in a PostgreSQL database modeled as RDF triples (`DB RDF`) or as a conventional schema (`DB Conventional`). The memory use is the peak of memory allocation for loading the necessary RDF/RDFS triples and build the view.



(a) Time to build the tree



(b) Memory use

Fig. 4. Comparing alternative schemes for generating ontology views

The running time measurements of Figure 4(a) show that `DB Conventional` is the fastest strategy. `RAM` is slightly slower than `DB Conventional` for large data sets, because of the burden of parsing RDF files, as opposed to efficiently taking instances from an indexed database via queries. `DB RDF` is by far the slowest alternative. This bad performance is probably due to the way RDF breaks the data about each instance – one RDF triple for each field value – leading to additional levels of indirection. Another advantage of `DB Conventional` over the other two strategies is that it uses a secondary index on the label values to order the position of sibling nodes in the ontology view. This ordering facilitates browsing the ontology view in the user's interface.

Figure 4(b) shows that the `RAM` strategy consumes the largest amount of memory. Both `DB Conventional` and `DB RDF` are more economical, because they do not require the construction of intermediate data structures in memory and take advantage of a database to load large sets of instances. `DB Conventional` is slightly more economical than `DB RDF`, perhaps due to Jena's housekeeping procedures for memory management. Therefore, the `DB Conventional` strategy is both fast and economical in terms of memory consumption.

7 Related Work

The Semantic Web [1, 13] foresees a new generation of Web based systems, taking advantage of semantic descriptions to improve the functionalities of current syntax-based data processing, and provide enhanced facilities in semantic aware open-ended information systems. Although much research effort has been directed to Semantic Web issues, few studies yet address engineering challenges, domain-specific issues, and the impact of ontology structure and ontology size, for example, on system design and performance.

The need for mechanisms to handle ontologies for semantic Web applications is recognized and addressed in [3, 8, 14, 7]. Dameron *et al.* [3] analyze some categories of ontology manipulation facilities necessary in a semantic Web infrastructure. They propose an architecture for offering facilities such as generation of ontology views, mapping between ontologies and reasoning, via Web services. Noy *et al.* [8] formalize the specification of ontology views by traversal – the same strategy employed in our prototype. Different mechanisms to express ontology views, such as using set operators on sets of classes and properties and restructuring hierarchies of classes are described in [14, 7]. Our traversal scheme is simpler and adherent to the POESIA approach and our applications.

8 Conclusions

This paper has considered implementation issues for loading, adapting and using domain ontologies for service discovery and composition on the Web. The main contributions are: (1) carrying out facilities adhering to the Semantic Web in a scientific application for the agricultural domain; (2) introducing the mechanism of ontological cover and a set of well defined relations among such covers to describe and recover services according to domain specific knowledge; (3) using ontology views with the acyclic restriction to enable the efficient manipulation of domain ontologies in applications. By using ontology views satisfying the acyclic restriction, we reduce a semantic problem (relating terms in an ontology), to a syntactic one (graph traversal or string comparison), without loss of semantic information. Our experimental results point out some shortcomings of current Semantic Web tools to handle large data volumes on producing ontology views, and we provide solutions to overcome these limitations. Though these results were presented in the context of a case study in agriculture, they apply to several domains and a wide class of applications that can benefit from the use of ontology views to manage data and services on the Web.

Acknowledgments. Authors from Campinas University were partially supported by Embrapa, CAPES, CNPq and projects MCT/PRONEX/SAI and CNPq/WebMaps. Authors from Georgia Tech were partially supported by two grants from the Operating Systems and ITR programs (CISE/CCR division) of NSF, by a contract from the SciDAC program of DoE, a contract from the PCES program (IXO) of DARPA, a faculty award and a SUR grant from IBM. Lauro Ramos Venncio has supplemented the experiments described on Section 6.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
2. F. Casati and U. Dayal (editors). Special issue on web services. *IEEE Data Engineering Bulletin*, 25(4), 2002.
3. O. Dameron, N. F. Noy, H. Knublauch, and M. A. Musen. Accessing and manipulating ontologies using web services. In *Intl. Semantic Web Conference (ISWC), Semantic Web Services Workshop*, 2004.
4. R. Fileto. *The POESIA Approach for Integrating Data and Services on the Semantic Web.* PhD thesis, Inst. of Computing, Campinas University, Brazil, 2003.
5. R. Fileto, L. Liu, C. Pu, E. D. Assad, and C. B. Medeiros. POESIA: An ontological workflow approach for composing web services in agriculture. *The VLDB Journal*, 12(4):352–367, 2003.
6. R. Fileto, C. B. Medeiros, L. Liu, C. Pu, and E. D. Assad. Using domain ontologies to help track data provenance. In *Proc. Brazilian Symposium on Databases*, pages 84–98, 2003.
7. A. Magkanaraki, V. Tannen, V. Christophides, and D. Plexousakis. Viewing the semantic web through RVL lenses. In *Intl. Semantic Web Conference (ISWC)*, pages 96–112, 2003.
8. N. F. Noy and M. A. Musen. Specifying ontology views by traversal. In *Intl. Semantic Web Conference (ISWC)*, pages 713–725, 2004.
9. N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with Protg-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2002.
10. PostgreSQL. `http://www.postgresql.org/` as of September 2003.
11. W3C's Resource Description Framework (RDF). `http://www.w3.org/RDF/` (as of October 2003).
12. W3C's RDF Query Language (RDQL). `http://www.w3.org/Submission/RDQL/` (as of November 2004).
13. W3C's Semantic web Activity. `http://www.w3.org/2001/sw/` (as of July 2004).
14. R. Volz and D. Oberlea nd R. Studer. Implementing views for light-weight web ontologies. In *Intl. Database Engineering and Applications Symp. (IDEAS)*. IEEE Computer Society, 2003.
15. K. Wilkinson, C. Sayers, and H. Kuno. Efficient RDF storage and retrieval in Jena2. In *Proc. Intl. Workshop on Semantic Web and Databases*, pages 131–150. Humboldt-Universitt, 2003.