# Semantic Web Technologies and Data Management

Li Ma, Jing Mei, Yue Pan
IBM China Research Laboratory
Bei Jing 100094, China

Krishna Kulkarni
IBM Software Group
San Jose, CA 95141-1003, USA

Achille Fokoue, Anand Ranganathan
IBM Watson Research Center
New York 10598, USA

## Introduction

The Semantic Web aims to build a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. It proposes to use RDF as a flexible data model and use ontology to represent data semantics. Currently, relational models and XML tree models are widely used to represent structured and semi-structured data. But they offer limited means to capture the semantics of data. An XML Schema defines a syntax-valid XML document and has no formal semantics, and an ER model can capture data semantics well but it is hard for end-users to use them when the ER model is transformed into a physical database model on which user queries are evaluated. RDFS and OWL ontologies can effectively capture data semantics and enable semantic query and matching, as well as efficient data integration. The following example illustrates the unique value of semantic web technologies for data management.

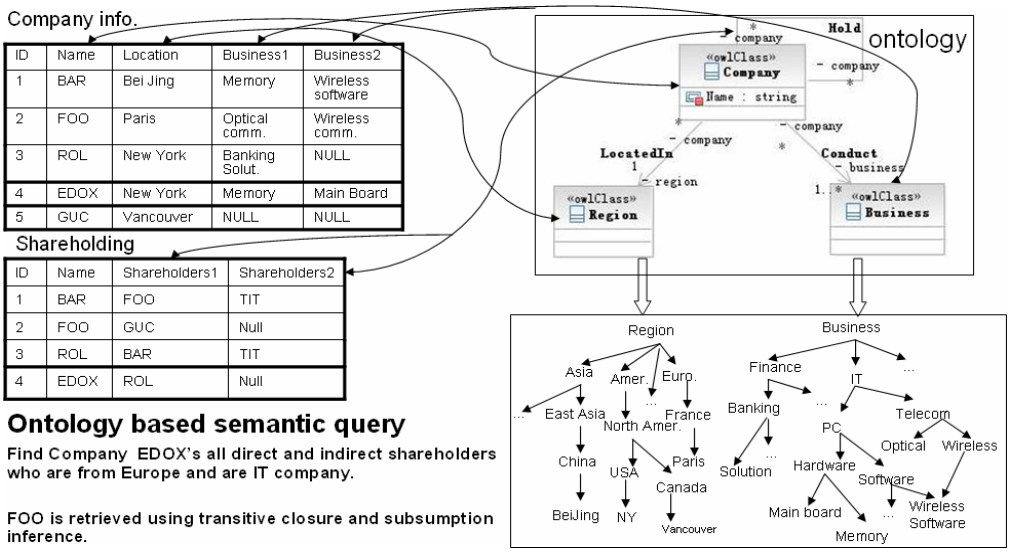


Figure 1. An example of ontology based data management

In Figure 1, we have two tables in a relational database. One stores some basic information of several companies, and another one describes shareholding relationship among these companies. Sometimes, users want to issue such a query "find Company EDOX's all direct and indirect shareholders which are from Europe and are IT company". Based on the data stored in the database, existing RDBMSes cannot represent and answer the above query. We want to retrieve shareholders in Europe, however, companies register their location in terms of cities. Similarly, we want to return shareholders in IT industry, however, companies register their business in terms of specific IT products. Also, we want to return both direct and indirect shareholders, which requires recursive look up where the number of iterations can not be known before evaluation. It is clear that the query is asked at a different level of concept granularity and a recursive look up is needed. We consider that the problem is not only caused by the use of different terms, but also that certain ontological knowledge is missing. By introducing two types of ontologies and leveraging inference implied by them, we can solve the above query effectively. The first type of ontologies encodes domain knowledge, such as the "Region" and "Business" ontologies which describe commonly-used classification trees for geography and industry, respectively. Such ontologies link specific data values to domain terms directly. Another kind of ontologies can be considered as a semantic representation of the data stored in databases. A simple example is the ontology shown in the top right of Figure 1, which describes entities and their relationships in the database and their formal definition, and thus provides a profile of the database information to users. In order to enable users to issue and evaluate queries based on the defined ontology for semantic representation, we need to map the data in relational tables to concepts and properties of the ontology. Figure 1 shows only a simplified (incomplete) mapping for illustration purpose. In summary, we can utilize ontologies to encode domain knowledge and provide a semantic representation to the information in databases, and further leverage ontology reasoning to solve the granularity mismatch between user queries and the data in databases, and answer semantic queries. Furthermore, ontologies which capture the semantics of the data will facilitate data integration as illustrated in [21].

Given that ontologies have unique values for semantic queries and information integration, it is worthy to investigate the problems of their use in data management. In this paper, we firstly survey the interoperability among relational data, XML and RDF, and that of SQL, XQuery and SPARQL. Then, we make a case study to illustrate the need and value of RDF representation and access to master data, and introduce some research work on Semantic Web from IBM. Finally, we discuss some open questions when publishing relational data as RDF resources and enabling ontology reasoning over legacy relational data.

# The Interoperability of RDF Data with Relational and XML Data

We observed two interesting findings from the history of relational and XML databases. Firstly, from query perspective, it is desirable that an existing query language is extended to retrieve a new type of data, allowing information from two formats to be correlated. For instance, use SQL to query both existing relational data and new XML data. Secondly, from data management perspective, it is valuable to express existing data with a newly-defined data model and query them with the corresponding query language. For example, publish relational data in an XML form for exchange purpose and use XQuery to retrieve them from databases. Here, we briefly summarize the potential interoperability of semantic web with relational and XML databases from data management and query perspectives.

Table 1 illustrates some extensions of existing query languages for access to new types of the data. It is well-known that SQL is extended to query both relational and XML data, namely SQL/XML, allowing SQL queries to create XML structures with a few powerful XML publishing functions. Assuming that relational databases serve as XML storage, XQuery queries need to be rewritten into SQL queries, as the XML extensions of commercial implementations provide. Alternatively, native XML databases have been developed and commercialized, which need to join results of XQuery and SQL. Similarly, with the development of RDF, SQL is also considered to support SPARQL queries, which has been implemented in a commercial database [2]. Currently, there are few native RDF databases and most RDF stores have been built on relational databases. Generally, RDF graphs are decomposed into triples, and the resulting RDF triples are shredded into a relational table of three columns (Subject, Property, Object). Rewriting queries from SPARQL to SQL becomes a challenge, utilizing well-developed SQL engines in a most effective manner. Specially, translating a complete SPARQL query into a single SQL statement is attractive, so that the generated SQL can be directly embedded as a sub-query into other SQL queries [4]. In addition, not losing beloved XML tools, RDF seems accessible by extending XQuery. On the one hand, an XML syntax specification for RDF, namely RDF/XML, has been defined in W3C recommendation, which implies RDF files as valid XML files. On the other hand, problems arise when uniting a graph structure of RDF with a tree structure of XML, and unfortunately, there is no canonical RDF/XML serialization. Consequently, XQuery with functional accessors is used to address the RDF graph and match parts of RDF statements. In particular, TreeHugger, using XQuery syntax to access RDF data, appears as another RDF query system based on path rather than graph matching.

Table 1. Extensions of existing query languages for the access to new types of the data

| **Host Language** | **Extended feature** | **Language Examples** | **Storage** | **Technical Requirements** | **Implementation Examples** |
|---|---|---|---|---|---|
| SQL | XML query | SQL/XML | XML in relational databases | Rewriting queries from XQuery to SQL | XML extension in commercial databases |
| | | | Native XML databases | Join results of XQuery & SQL | Commercial native XML stores |
| SQL | RDF query | SQL table function | RDF in relational databases | Rewriting queries from SPARQL to SQL | Commercial RDF stores |
| | | | Native RDF databases | Join results of SPARQL & SQL | N/A |
| XQuery | RDF query | XQuery with Functional Accessors; | RDF in XML serialization | Normalized XML representation of RDF; SPARQL implementation using XQuery | TreeHugger |
| | | | Native RDF databases | Join results of SPARQL & XQuery | N/A |

Newly-defined data models and query languages can implement their values over legacy data sources by publishing and query rewriting technologies. The following table summarizes such work. As the support of XQuery over relational data is well-known, we address the problem of SPARQL access to relational and XML databases. We can put an RDF cap for a relational data source, and a representative is D2RQ mapping [5], which treats non-RDF relational databases as a virtual RDF graph by generating URI patterns in ClassMap and PropertyBridge. Similarly, SquirrelRDF [8] provides the database mapping which could be automatically configured so that tables become RDF classes and columns become RDF properties with the table class as their domain. Alternatively, OpenLink Virtuoso [6] renders relational schema into RDF by assigning a property URI to each column and an rdf:type property for each row linking it to an RDF class URI corresponding to the table. Others, allowing non-RDF relational data to be queried using SPARQL, include SPASQL (SPARQL support in MySQL) [9] and Relational.OWL (a data and schema representation format based on OWL) [7]. Based on the built mapping between relational schema and ontologies, SPARQL queries can be rewritten into SQL statements to retrieve relational data. Note that, SPARQL seems to be close to recommendation; however the formal semantics of SPARQL is still an open issue, resulting in different methods for SPARQL support. Perez et al. proposed one [11], regarding AND as the SQL join, UNION as the SQL union, OPT as the SQL left outer join, etc. Recently, refined and extended versions are presented in [12], having three variants, namely bravely joining, cautiously joining and strictly joining semantics, where the bravely joining semantics coincides with [11]. In particular, based on the three semantic variants, [12] also provides translations from SPARQL to Datalog with negation as failure, which might serve straightforwardly to implement SPARQL within existing rule engines. As some existing applications have used XML databases to manage their data, supporting SPARQL access to XML databases is also attractive for integration. But now, there is little such work. A related work is GRDDL (Gleaning Resource Descriptions from Dialects of Languages) which provides the mechanism to extract RDF data from general-purpose XML documents. **Because of the semantics differences between different query languages, we believe attempts to extend SQL and XQuery to support**

**SPARQL would involve considerable complexity. Hence, we advocate focusing research efforts on publishing and accessing relational (and XML) data as RDF data and exploiting SPARQL for semantic query and integration.**

Table 2. Publishing and accessing legacy data using new data models and query languages.

| Language | Data Sources | Technical Requirements | Implementation Examples |
|---|---|---|---|
| XQuery | Relational databases | Publish relational data as XML<br>Rewrite XQuery to SQL | Commercial databases |
| SPARQL | Relational databases | Publish relational data as RDF<br>Rewrite SPARQL to SQL | D2RQ mapping and D2R Server<br>Virtuoso; SPASQL |
| SPARQL | XML databases | Publish XML data as RDF<br>Rewrite SPARQL to XQuery ; | N/A |

## A Case Study: RDF Representation and Access to Master Data

Master data, as the core business entities a company uses, refers to lists or hierarchies of customers, suppliers, accounts, products, or organizational units [16]. In this scope, Product and Customer Information play a very important role since their accurate management is becoming critical for modern enterprises. They enable companies to centralize, manage and synchronize all product and customer information with heterogeneous systems and trading partners. The most critical challenge is the need to build a common master model flexible enough to deal with business changes, and expressive enough to represent the semantics of master data.

To enhance IBM master data management (MDM) solutions (Websphere Product Center and Websphere Customer Center) [15], we developed semantic technologies for Product Information Management (PIM) and Customer Data Integration (CDI), respectively. Here, we would like to highlight the value of semantic web technologies for MDM and brief completed and ongoing work. As introduced in our previous work [1], the advantages of OWL ontologies for product information include followings:

- As based on RDF, OWL uses the concept of Universal Resources Identifiers (URIs) as Web-based identification scheme. It firstly allows one to refer to industry specific or external ontologies; and on the other hand it allows synchronization of product information management utilities to other core business entities, such as those in customer data integration (CDI).
- OWL allows the definition of **richer properties and relationships**. Object Properties can be defined as symmetric, functional, inverse functional, or transitive. Object Properties are then suitable to describe complex relationships among products and between products and other entities in product information.
- The expressivity of OWL allows the definition of **logical classes** (*intersection*, *union* and *complement* operators), which enables automatic classification for product items. For instance, new product categories can be defined as the intersection of two others: *smartphones* products, which gather characteristics of both PDA and phones, are a good example. Any product which is simultaneously a PDA and a phone is then a *smartphone*.
- OWL restrictions can define **dynamic categories** which do not exist in the pre-designed category hierarchy and are specified by users at query time. It can represent complex and potentially evolving categories. For example, using *Minimum cardinality restriction*, it is possible to define an “outdated products” category which gathers all products replaced by at least one other product. Items of dynamic categories can be retrieved using OWL ontology reasoning.

Details of RDF representation of PIM can be found in [1]. Since IBM PIM system currently uses technologies similar to the triple store for storage (item-property-value), we support SPARQL queries over PIM storage easily by reusing SPARQL2SQL query translation technologies developed in [17]. The query rewriting method translates a SPARQL query into a single SQL statement, utilizing well-developed SQL engines in a most effective manner.

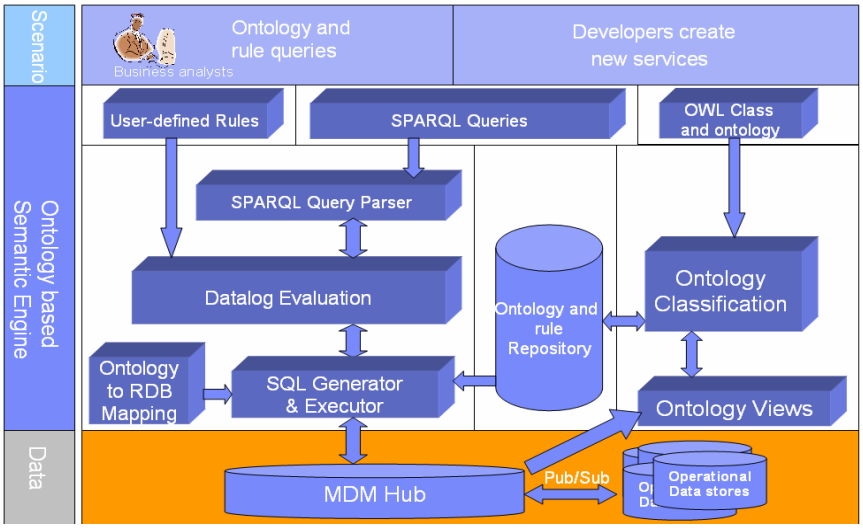


Figure 2. Conceptual Architecture for SPARQL Queries over the CDI system

The advantages of OWL ontologies for customer information are similar to those for product information. Representing and discovering various relationships among customers has a very high value for the CDI, which is enabled by ontology and rule reasoning. Different from the PIM system, IBM CDI system makes use of object-oriented database schema for storage. Each entity of the CDI model owns a separate table to store corresponding instances. So, we need a mapping to link the CDI data with the OWL ontology generated and enriched from the CDI logical model. We proposed the following conceptual architecture to develop a POC system for RDF Access to the IBM CDI system. Note that the ontology view in the bottom right of Figure 2 is in fact a virtual representation of the CDI data in SOR's schema form [4], over which IBM SHER engine for ontological reasoning can work directly. In the future, we will support SHER engine over a D2RQ like mapping [5], and thus replace the ontology views component. Besides the mapping between relational DB and ontology, it is critical to construct an appropriate RDF representation (ontology) for relational data. The challenge in using domain knowledge in ontologies effectively is in crafting "integrating" ontologies that tie the domain knowledge in ontologies with other ontologies that may be used to model the relational data in the database. For example, in Figure 1, a crucial piece in answering the query is the ontology in top right, which ties the data in the relational DB with the domain ontologies describing regions and businesses. We would need to come up with guidelines and best practices for developing these ontologies

## IBM's Semantic Web Tools and Systems

Here, we briefly introduce some IBM's ontology tools and systems related to RDF Access to relational data. IODT [10] is a toolkit for ontology-driven development, including EMF Ontology Definition Metamodel (EODM) and an OWL Ontology Repository (named SOR). EODM is derived from the OMG's Ontology Definition Metamodel (ODM) [13] and implemented in Eclipse Modeling Framework (EMF). It is the run-time library that allows the application to put in and put out an RDFS/OWL ontology in RDF/XML format; manipulate an ontology using Java objects; call an inference engine and access inference results; and transform between ontology and other models. SOR [4] is an OWL ontology storage and query system on the relational DBMS. It supports Description Logic Program (DLP), a subset of OWL DL, and SPARQL query language. SHER reasoner [14] uses a novel method that allows for efficient querying of SHIN ontologies with large ABoxes stored in databases. Currently, this method focuses on instance retrieval that queries all individuals of a given class in the ABox. It is well known that all queries over DL ontologies can be reduced to consistency check, which is usually checked by a tableau algorithm. SHER groups individuals which are instances of the same class into a single individual to generate a summary ABox of a small size. Then, consistency check can be done on the dramatically simplified summary ABox, instead of the original ABox. It is reported in [14] that SHER can process ABox queries with up to 7.4 million assertions efficiently, whereas the state of art reasoners could not scale to this size.

As described in the example shown in Figure 1, to enable semantic queries over existing data sources, we need to store and leverage ontologies representing domain knowledge. SOR could be used to manage such ontologies. Similarly, in the CDI case, we need an ontology repository to cache and materialize some inference results for performance improvement. In general, an RDF store, such as SOR, could be used to store domain knowledge or part of reasoning results for RDF access to relational databases. Obviously, SHER engine could be used for scalable ontological reasoning for SPARQL queries over relational databases. The system described in [22] takes an ETL (Extract-Transform-Load) approach, where the relational data in the database is extracted, transformed into RDF triples based on a set of domain ontologies and mapping rules, and loaded into SOR. This system also provides mechanisms to handle updates to the relational database as well as to the ontologies.

## Discussions

There are three key steps when exposing relational data as RDF data, creating an RDF Representation (ontology) of the relational data, building a mapping between the relational database and ontology, and rewriting SPARQL queries to retrieve the relational data. Recently, progresses have been made in these three aspects, but following challenges need to be paid more attentions.

- URI Generation for the relational data: Recalling that RDF resources are identified by URI, efforts in choosing good URIs for relational data are worthy. Tim Berners-Lee ever uses a term, "cool", for URIs designed with simplicity, stability and manageability in mind [18], and a recent article [19] presents two strategies, called 303 URIs and hash URIs. Another solution, proposed in [20], ends up with three URIs related to a single non-information resource, i.e., an identifier for the resource, an identifier for a related information resource suitable to HTML browsers with a web page representation, and an identifier for a related information resource suitable to RDF browsers with an RDF/XML representation. The problem of URI generation deserves more efforts. Another related problem is instance-mapping, i.e., how a particular data element in a cell may be mapped to the URI of an individual in an OWL ontology.
- N-Ary Relationship Representation and Query: Our experiences show that real applications often include a large number of N-Ary relationships. But, ontologies are limited to represent N-Ary relationships and SPARQL does not have built-in constructs for them. In practice, therefore, we usually use RDFS classes to express N-Ary relationships and rules to specify reasoning on them, instead of built-in constructs, like transitive property. A graceful way to represent N-Ary Relationship in ontologies is desirable, using the best practices documented at http://www.w3.org/TR/swbp-n-aryRelations/ as a starting point.
- Representation of RDB Schema Constraints in Mapping: As we know, an RDB schema uses a set of built-in vocabularies to represent table structure. For instance, the term UNIQUE denotes that the data value of a column is both NOT NULL and distinct. Such information is highly valuable for query rewriting (and thus for information integration) and should be included in the mapping from RDB to RDFS/OWL ontology. Unfortunately, existing mapping techniques do not capture such information. So, it is desirable to develop a powerful standard mapping language in the future.

- Effective Query Rewriting and Optimization: Translating SPARQL queries to SQL queries is widely studied in RDF data management. Filter expressions, which restrict the graph pattern matching solutions to express specific requirements on results, often consist of multiple functions and operators, and a filter operator may have different behaviors on different operands. Existing systems, such as Sesame and Jena2, usually adopt memory-based methods to evaluate SPARQL filter expressions, rather than directly leveraging the database query engine. Lu et al. [17] proposed an effective method to translate a SPARQL query with filter expressions into a single SQL query, making use of optimized database query engines as much as possible. Effective query rewriting and optimization based on the mapping from RDB to ontology need further investigation.
- Reasoning Issues: One advantage of using OWL ontologies is that DL reasoning can be used to return additional results to queries. However, this brings additional issues, particularly when more expressive logics are used. One issue of importance is reconciling the closed world nature of databases with the open world nature of Description Logic reasoning.
- Performance and Security Issues: A relational data source supports a specific kind of applications. When we expose such a data source as an RDF source, we need to consider access control issue and performance impact. SPARQL queries and ontology reasoning on a data source may need expensive database operations and thus impact the performance of existing applications over the same data source. So, it is valuable to study such impact in depth.

From modeling perspective, we think there are three key issues to implement semantic web technologies enabled data management and integration. The first is on the semantic web data representation, such as RDFS and OWL specifications. The second is on the ontology mapping which defines correspondences among different ontologies. The third issue is on the mapping between ontology and underlying data sources (such as relational databases and XML stores). Considering that most existing data is stored in relational databases, it is highly valuable to expose relational data in an RDF format with semantics defined in ontologies. Currently, W3C has recommended RDFS and OWL as specifications, and is organizing OWL 1.1 working group to discuss problems when applying OWL in practice and thus make corresponding extensions to OWL. Research efforts on the 2nd and 3rd issues are significant, but without specification support yet. One reason, we think, is that OWL1.1 extension is ongoing and may affect the latter two issues seriously. So, when to start working on standards for the mapping among ontologies and RDF representation for relational data may depend on the maturity of the extended OWL specification. **In summary, we think that:**

- **Expressing and accessing relational data as RDF resources through a mapping between database and ontology is highly valuable;**
- **There are still many interesting research problems that need to be solved. We advocate setting up an incubator group or a working group to identify and address all such problems before embarking on any standardization.**

## References


1. J-S. Brunner, L. Ma, C. Wang, L. Zhang, Y. Pan, K. Srinivas. Explorations in the use of Semantic Web Technologies for Product Information Management. In Proc. of WWW 2007, pp. 747 - 756.
2. C. Murray, N. Alexander. Oracle Spatial Resource Description Framework (RDF), 10g Release 2 (10.2), 2005.
3. J. Melton. SQL, XQuery, and SPARQL: Making the Picture Prettier. In Proc. Of XML 2006.
4. J. Lu, L. Ma, L. Zhang, J-S. Brunner, C. Wang, Y. Pan, Y. Yu. SOR: A Practical System for OWL Ontology Storage, Reasoning and Search. In Proc. of VLDB 2007, to appear.
5. D2RQ. http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/
6. Virtuoso. http://virtuoso.openlinksw.com/wiki/main/Main/VOSSQLRDF
7. Relational.OWL. http://sourceforge.net/projects/relational-owl/
8. SquirrelRDF. http://jena.sourceforge.net/SquirrelRDF/
9. SPASQL: SPARQL Support In MySQL. http://www.w3.org/2005/05/22-SPARQL-MySQL/XTech
10. IBM Integrated Ontology Development Toolkit. http://www.alphaworks.ibm.com/tech/semanticstk
11. J. Pérez, M. Arenas, C. Gutierrez. Semantics and Complexity of SPARQL. In Proc. of ISWC 2006, pp. 30 - 43
12. A. Polleres. From SPARQL to Rules (and back). In Proc. of WWW 2007, pp. 787 – 796
13. Ontology Definition Metamodel (ODM) Request for Proposal, OMG Document: ad/2003-03-40. http://www.omg.org/cgi-bin/doc?ad/06-05-01.pdf.
14. J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, E. Schonberg, K. Srinivas. Scalable Semantic Retrieval Through Summarization and Refinement. In Proc. of AAAI 2007, pp. 299 - 304.
15. IBM Multiform Master Data Management. http://www-306.ibm.com/software/data/ips/products/masterdata/
16. H.D. Morris, D. Vesset. Managing Master Data for Business Performance Management: The Issues and Hyperion's Solution. IDC white paper, 2005.
17. R. Lu, F. Cao, L. Ma, Y. Yu, Y. Pan. An Effective SPARQL Support over Relational Databases. In Proc. of SWDB-ODBIS07 co-located with VLDB 2007, to appear.
18. T. Berners-Lee. Cool URIs don't change. http://www.w3.org/Provider/Style/URI
19. L. Sauermannh, R. Cyganiak, M. Volkel. Cool URIs for the Semantic Web. DFKI Technical Memo TM-07-01
20. C. Bizer, R. Cyganiak, T. Heath. How to Publish Linked Data on the Web. http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/
21. N. Noy. Semantic Integration: A Survey of Ontology-Based Approaches. SIGMOD Record, 33(4), 2004
22. A. Ranganathan, Z. Liu. Information Retrieval from Relational Databases using Semantic Queries. In Proc. of ACM CIKM 2006, pp. 820 - 821