SQL Databases are a Moving Target
Position Paper for W3C Workshop on RDF Access to Relational Databases

Juan F. Sequeda, Syed Hamid Tirmizi and Daniel P. Miranker
The University of Texas at Austin
Department of Computer Sciences
{jsequeda, hamid, miranker} @ cs.utexas.edu

Abstract

Our position is founded on two assumptions. First, we assume that the SQL data definition language (SQL-DDL) is capable of encoding substantial domain semantics, albeit not in ways syntactically accessible to inference engines. Second, the layered architecture of the Semantic Web, a.k.a. the layer cake*, transcends the Semantic Web. In other words, we assume that the hierarchy of increasingly expressive Semantic Web languages embody a generalizable organizational principle. We do not venture to draw parallels to computational hierarchies. Rather, we cast SQL-DDL into a Semantic-Web like layer cake with the intention to make apparent SQL's ability to embody semantics and to rationalize the approach of mapping relational databases to the Semantic Web. Even though proportionally few applications exploit the full expressive power of SQL-DDL we show that it is important to develop relational database to Semantic Web mappings that anticipate the full expressive power of SQL.*

Introduction

It is commonly understood that there are more dynamically generated web pages than there are static web pages. It follows that most of the data on the Internet is sourced from relational databases, and thus the success of the Semantic Web hinges on effective mappings of relational databases and their content to the Semantic Web. Likewise, the labor required to make a database available as Semantic Web content usually rests with the database administrator. It is rare that integrating a database with the Semantic Web accrues obvious benefit to the owners of a database. Thus, organizationally, integrating a database with the Semantic Web is rarely a priority. Hence, it is imperative for the community to develop fully automated methods for bridging relational database content and the Semantic Web.

Broadly stated, there are two architectural approaches to integrating databases with the Semantic Web.

1. Relational Database to Ontology Mapping: Given the centrality of ontologies to the Semantic Web, most efforts are concerned with wrapper systems that connect the database and its schema to a domain ontology. The ontology may be either a shared global or local ontology. Depending on the aggressiveness of the individual project, the construction of the wrapper may range from strictly manual wrapper programming to sophisticated mapping languages and environments that may even contribute automatic inferences to help define some of the mapping systems. [1, 2, 3, 4, 5, 6, 7, 8, 9]
2. Direct Mapping of a Relational Database to the Semantic Web: In a direct mapping an external ontology is not considered. The database content data and/or its schema are used to export the database to a Semantic Web representation (e.g. RDF or OWL). For example D2RQ system [3] translates database tables of n-tuples to RDF triples. Thus, data in tables defined in the relational schema of Figure 1a, would be translated into RDF triple form equivalent to the illustration in Figure 3. A number of projects have undertaken the direct mapping approach [10, 11, 12, 13, 14].

We admit that the scope and frequency of success of direct mapping is very unlikely to achieve the scope of description ontology mapping methods can achieve. However, direct-mapping methods are intrinsically completely automated. Since the community is still in early stages of investigation, we feel that if we simply clearly define, (by pushing), the limits of direct mapping larger benefits will accrue. Thus, we feel that it is imperative to fully explore this direction.

Increasingly Empowered SQL Data Models

SQL is a moving target. Over the last two decades, in an ongoing sequence of expanding standards, (SQL 86-89, 92, 99, 2003) continues to evolve. SQL is organized in parts, the data definition language (SQL-DDL), the query language and the data manipulation language. SQL-DDL is a declarative language for defining both the logical and physical representation of data in a database. Taking some literary license we will provide an overview of the DDL features as they have evolved and show they run parallel with the expressive hierarchy reflected in the Semantic Web layer cake.

Prior to the creation of relational query languages the relational model was largely concerned with the representation of data (Figure 1a) as n-ary relations and the correctness of syntactic transformations relational algebra [15]. Data types for the columns were practically an afterthought. The central concern was the theory of normal forms. A primary goal was to reduce storage overhead.

The first SQL standard, SQL86-89 was limited. For the purpose of this discussion, a standard set of data types were defined, abbreviated formal relational notations were replaced full English words, and the now familiar accoutrements of human readable computer languages added (Figure 1b). The often heard claim that SQL databases do not provide for the encoding of data semantics is apt for this version of SQL [16].

SQL 92 added data integrity constraints: CHECK, PRIMARY KEY, FOREIGN KEY and UNIQUE [17]. This was the first approach to offer domain semantics by connecting relations and permitting us to know beforehand what values are allowed. In conjunction with the basic table definition of SQL 88, this version of SQL is the current one that is used today. See Figure 1c.

One of SQL 99's main components was Triggers [18]. Triggers are forward-chaining rules whose predicate includes an update event on a table. Triggers are often used to maintain correctness and to implement heterogeneous data integration [19]. Figure 1d is a trigger that maintains an invariant on the database concerning salary inversion. Even though SQL 2003 introduces XML-related features, it does not add any specific semantics components to the DDL.

a) Relational Model Before SQL	`Employee (name, age)`
b) Table Definition SQL 86-89	`CREATE TABLE employee (name VARCHAR(100), age INTEGER)`
c) Constraints SQL 92	`CREATE TABLE employee(` `name VARCHAR(100) PRIMARY KEY,` `salary INTEGER NOT NULL,` `type CHAR(8) CHECK(type IN ('TEMP','FULLTIME','CONTRACT'))` `dept_name FOREIGN KEY (dept) REFERENCES department (name))` `CREATE TABLE department(` `name VARCHAR(100) PRIMARY KEY)`
d) Triggers SQL 99	`CREATE TRIGGER sal_adjustment` `AFTER UPDATE OF salary ON employee REFERENCING OLD AS OLD_EMP NEW AS NEW_EMP` `FOR EACH ROW WHEN (NEW_EMP.SALARY > (OLD_EMP.SALARY *1.20))` `BEGIN ATOMIC SIGNAL SQLSTATE '75001'('Invalid Increase: > 20%'); END`

Figure 1. Evolution of Relational DDL

SQL Layer Cake

Knowing the historical context, it is easy to decompose SQL-DDL into a stack representing increasing expressive power and, perhaps not by coincidence, one that corresponds to the Semantic Web stack.

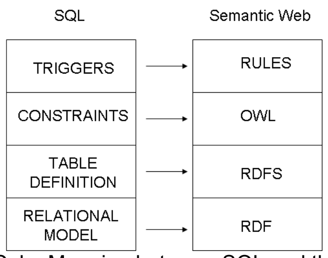

Figure 2. Layer Cake Mapping between SQL and the Semantic Web

As we seek to translate more of SQL to the Semantic Web, we find ourselves in a position where we have to examine the correspondent use of each Semantic Web layer. We find that this is driven as much by the amount of expressive power of the Semantic Web we want to involve in a system as it is the scope of the SQL definition we wish to capture. For example, if a database application makes full use of SQL92 syntax to specify a number of constraints on the data, one is inclined to map the database schema to OWL constructs (`owl:Class`, `owl:DatatypeProperty`, `owl:ObjectProperty`). If the ultimate goals are fulfilled by only RDF, then utilizing the Relational Model is enough, independent of the actual use of SQL-DDL in a database. In other words the constraints may be ignored and only the table definitions, as could be expressed in SQL89, could be mapped to RDFS constructs (`rdfs:Class`, `rdf:Property`).

In order to show the evolving semantics of SQL and the relationship between each layer of SQL and the Semantic Web, we have devised examples for each layer mapping where we denote that each super layer offers more semantics than the previous one. Observe how the column name "age" evolves through each layer.

Relational Model and RDF

The first layer we encounter is the Relational Model, which predates SQL. This layer only expresses the syntactic structure of the data stored in a relational database; therefore we can map it only to the data layer of the Semantic Web: RDF. To export the content in a relational database to the Semantic Web, it is necessary to create an RDF representation of this content. A number of systems in [10, 11, 12, 13, 14] have exploited this approach.

Likewise, RDF expresses data in a similar way representing it as a labeled graph in the form of a Subject-Predicate-Object statement. The n-tuples are similar to the triple statement of RDF. An example of the RDF representation is in the Appendix. In this example, the column name "age" denotes the edge of the graph.

T1: Employee (Juan, 21)

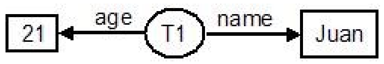

Figure 3: Figure 1a as an RDF Diagram

Table Definition and RDFS

The relational model offers sufficient semantics to create relations between the data, but unfortunately it does not consist of domain semantics. The first version of SQL (SQL89), introduced the table definition, which creates a relational schema for the data. Each column has a specific data type. Likewise, RDFS is a schema for RDF data which creates classes and properties.

An equally valid representation of a table definition is an n-dimensional graph, where n is the number of column names. This graph representation is similar to RDFS, where it established a schema to represent data in RDF triple form. Therefore, by utilizing this version of SQL, its domain semantics can by mapped only to the RDFS layer. RDFS can represent the table definition (Figure 1b) by having a class Employee with properties name and age. An example of the RDFS format is in the Appendix. In this example, the column name "age" becomes and `rdf:Property` as shown in Figure 4b.

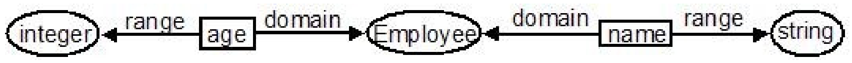

Figure 4a. Figure 1b has an RDFS Diagram

```
<rdf:Property rdf:ID="age">
      <rdfs:comment>Age of Employee</rdfs:comment>
      <rdfs:domain rdf:resource="#employee"/>
      <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
   </rdf:Property>
```

Figure 4b. RDFS format of "age" RDF Property

Constraints and OWL

Climbing the Semantic Web layer cake, more semantics are encountered. OWL offers expressiveness that can not be exploited in RDFS. Likewise, SQL99 and in conjunction with elements of SQL92, contains new components that offer more domain semantics. By utilizing PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE and CHECK, domain semantics are more explicit. These semantics are richer than the basic table definition and therefore should be mapped to a higher level in the Semantic Web layer cake. We have encountered that the use of SQL with all the possible constraints will map directly to OWL (hence the use of

OWL class instead of RDFS class). Even though RDFS and OWL are both ontology languages, OWL is more expressive due to the fact that it can represent specific type of properties and restrictions. The main similarity between database constraints and OWL is the possibility of using the database's referential constraint to establish relationships between tables as `owl:ObjectProperty` that connects objects. PRIMARY KEY, UNIQUE and NOT NULL implies `owl:FunctionalProperty` and the enumerated CHECK constraint implies `owl:oneOf` (this will be discussed in the next section).

Following the example in Figure 1c, the schema and its constraints can be represented in OWL. In a general assumption, a relation can be considered an OWL class except if the relation is a weak entity, therefore it is an OWL object property; all attributes that are referential constraints are OWL object properties, whose domain is the current relation and range is the referenced relation, and attributes that are not referential constraints are OWL data type properties. In OWL, the "age" column name becomes an `owl:DatatypeProperty`, as shown in Figure 5. An example of the OWL ontology is in the Appendix.

```
<owl:DatatypeProperty rdf:ID="age">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Employee"/>
    <rdfs:range rdf:resource="&xsd;int"/>
</owl:DatatypeProperty>
```

Figure 5. "Age" attribute of Figure 1c in OWL format

Triggers and Rules

The latest version of SQL (SQL 2003) introduces a new semantic component: triggers. This new component encodes business rules that will guarantee the integrity of data while it is being manipulated in the database. The rules in the Semantic Web have the same objective. Therefore we consider that this mapping is an open discussion.

Discussion

Other specific discussion points are relevant when coming to creating OWL from SQL. We have identified two main points: mapping to `owl:someValuesFrom` or `owl:allValuesFrom`, and the semantics encoded in SQL's CHECK constraint.

Specific examples often appear to offer more semantics than is encoded in a SQL model. Since domain semantics are often obvious to people, independent of their representation, development of mappings of SQL to the Semantic Web are often equivocal. A confusing example is the use of `owl:someValuesFrom` and `owl:allValuesFrom`. In Xu et al.'s approach [20], the `owl:allValuesFrom` restriction is applied to every class and their object properties. Even though the reason was not explained, it can be understood that because the source is an Entity-Relationship model, the semantics are more explicit than SQL and therefore it can be assumed that all values from one class are the properties of another class. In SQL it is different. If we consider for example:

```
A(id, x)    B(id, y)    C(A_id, B_id)
```

We can ask and answer the following questions: Can we say that A and B are classes? Can we say that C represents an object property? An answer, to both, is yes. Can we say that B can only be connected to A using object property? Can we say anything about the cardinality of this relation? We still can not answer the previous two questions.

Another component that deserves discussion is CHECK. This constraint embodies semantics and rules at the same time. OWL is not expressive enough to represent all the possibilities that the semantics of CHECK offers. The only CHECK constraint that can be applied to OWL is the enumerated constraint which can be represented using `owl:oneOf` and `rdf:List`.

In our layer cake mapping, we establish a one-to-one correspondence between each layer. In the simplest sense, a table is a simple class where a class is mapped to RDFS (as shown in our layer cake, Figure 1b and Figure 4). But when constraints are added, the tables ought to be OWL classes (Figure 1c and Figure 5) since the constraints map to OWL constructs and it becomes necessary to have OWL classes to leverage the semantics encoded in these constraints.

In conclusion, a key point of the Semantic Web in this context is to facilitate heterogeneous data integration. Whereas the Semantic Web began with an established hierarchy of semantics, the changes in SQL standards are incrementally increasing their semantic power. Our observation of the column name “age” demonstrates that a single component of SQL can be mapped to different layers of the Semantic Web. A similar situation occurs in Krishnamurthy et al. [21] where they argue that data integration is best expressed in higher-order logic. A canonical mapping of SQL constructs to the Semantic Web is needed to avoid these complexities. To facilitate data integration, it is needed that the community shares the greatest common denominator when it comes to SQL mapping to the Semantic Web, which in this case is OWL.

This research is funded in part by the National Science Foundation grant IIS-0531767.

References

[1] Rodriguez, J. B. and Gómez-Pérez, A. 2006. Upgrading relational legacy data to the semantic web. In *Proceedings of the 15th international Conference on World Wide Web*. WWW '06
[2] Barrasa J., Corcho O., Gómez-Pérez A. (2004): R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. A. Second Workshop on Semantic Web and Databases (SWDB2004)
[3] Bizer, C. Cyganiak, R. 2006. D2R Server - Publishing Releational Databases on the Semantic Web (Poster). Poster at the *5th International Semantic Web Conference* (ISWC2006)
[4] An, Y. Mylopoulos, J. Borgida, A. 2006. *Building Semantic Mappings from Databases to Ontologies.* In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)
[5] Xu, Z., Zhang, S., and Dong, Y. 2006. Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. In *Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence*
[6] Nyulas, C. O'Conner, M. Ty, S. 2007. DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé. In Proceedings of 10th International Protégé Conference
[7] Chen, H. Wang, Y. Wang, H. Mao, Y. Tang, J. Zhou, C. Yin, A. Wu, Z. 2006. Towards a Semantic Web of Relational Databases: a Practical Semantic Toolkit and an In-Use Case from Traditional Chinese Medicine. In Proceedings of *5th International Semantic Web Conference* (ISWC2006)
[8] Dou, D. Pan, J. Qin, H. LePendu, P. Towards Populating and Querying the Semantic Web. In *Proc. of the ISWC2006 Workshop on Scalable Semantic Web Knowledge Base Systems*.
[9] Korotkiy, M. Top, J. From Relational Data to RDFS Models. In Proceedings of the *International Conference on Web Engineering*. Munich, 2004.
[10] de Laborda, C. P. and Conrad, S. 2005. Relational.OWL: a data and schema representation format based on OWL. In *Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling*
[11] Svihla M., Jelínek I. 2004. Two Layer Mapping from Database to RDF. In *Proceedings of Electronic Computers and Informatics (ECI)*, Slovakia, Kosice
[12] Laclavik, M. 2006. RDB2Onto: Relational Database Data to Ontology Individual Mapping In*: Tools for Acquisition, Organisation and Presenting of Information and Knowledge*. P.Navrat et al. (Eds.),
[13] Bizer, C. Seaborne, A. 2004. D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs. Poster at *3rd International Semantic Web Conference* (ISWC2004)
[14] Bizer, C. 2003. D2R MAP - A Database to RDF Mapping Language. *The Twelfth International World Wide Web Conference* (WWW2003)
[15] Maier, David. The Theory of Relational Databases. Computer Science Press 1983
[16] Pratt, Philip J. A Guide To SQL. Boston: Boyd & Fraser Publishing Company, 1990.
[17] Pratt, Philip J. A Guide To SQL. 3rd ed. Boston: Boyd & Fraser Publishing Company, 1995.
[18] SQL3 (ISO-ANSI Working Draft) http://www.inf.fu-berlin.de/lehre/SS94/einfdb/SQL3/sqlindex.html
[19] Ceri, S. Widom, J. 1993. Managing Semantic Heterogeneity with Production Rules and Persistent Queues. In *Proceedings of the 19th international Conference on Very Large Data Bases*
[20] Xu, Z. Cao, X. Dong, Y. Su, W. 2004. Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies. In Proc of Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, (PAKDD2004)
[21] Krishnamurthy, R., Litwin, W, Kent, W. 1991. Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the 1991 ACM SIGMOD international Conference on Management of Data.*

Appendix

Relational Model and RDF

```
Employee(Juan, 21)

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:ex="http://www.example.com/#">
     <rdf:Description rdf:about="http://www.example.com/employee">
             <ex:name>Juan</ex:name>
             <ex:age>21</ex:age>
     </rdf:Description>
   </rdf:RDF>
```

Table Definition and RDFS

```
CREATE TABLE employee (
       name VARCHAR(100),
       age INTEGER)

<?xml version="1.0"?>
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

   <rdfs:Class rdf:ID="employee">
     <rdfs:comment>Employee</rdfs:comment>
     <rdfs:subClassOf
          rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource"/>
   </rdfs:Class>

   <rdf:Property rdf:ID="name">
      <rdfs:comment>Name of Employee</rdfs:comment>
      <rdfs:domain rdf:resource="#employee"/>
      <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
   </rdf:Property>

<rdf:Property rdf:ID="age">
      <rdfs:comment>Age of Employee</rdfs:comment>
      <rdfs:domain rdf:resource="#employee"/>
      <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
   </rdf:Property>
   </rdf:RDF>
```

Constraints and OWL

```
CREATE TABLE employee(
       employee_id INTEGER PRIMARY KEY,
      employee_ssn VARCHAR(11) UNIQUE,
       employee_name VARCHAR(100) NOT NULL,
      employee_salary INTEGER NOT NULL,
      employee_type CHAR(8) CHECK ( employee_type IN ('TEMP', 'FULLTIME', 'CONTRACT'))
       dept INTEGER FOREIGN KEY (dept) REFERENCES department (dept_id))

CREATE TABLE department(
       dept_id INTEGER PRIMARY KEY,
       dept_name VARCHAR(100) NOT NULL,
       manager INTEGER FOREIGN KEY (manager) REFERENCES employee (employee_id))


<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1189059398.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1189059398.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="Department"/>
    <owl:ObjectProperty rdf:ID="dept">
        <rdfs:domain rdf:resource="#Employee"/>
        <rdfs:range rdf:resource="#Department"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="dept_id">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Department"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="dept_name">
        <rdfs:domain rdf:resource="#Department"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="Employee"/>
    <owl:DatatypeProperty rdf:ID="employee_id">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Employee"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="employee_salary">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Employee"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="employee_ssn">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Employee"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="employee_type">
        <rdfs:domain rdf:resource="#Employee"/>
        <rdfs:range>
            <owl:DataRange>
                <owl:oneOf>
                    <rdf:List>
                        <rdf:first rdf:datatype="&xsd;string">Temp</rdf:first>
                        <rdf:rest>
                            <rdf:List>
                                <rdf:first rdf:datatype="&xsd;string">
                                        Fulltime
                                 </rdf:first>
                                <rdf:rest>
                                    <rdf:List>
                                        <rdf:first rdf:datatype="&xsd;string">
                                              Contract
                                        </rdf:first>
                                        <rdf:rest rdf:resource="&rdf;nil"/>
                                    </rdf:List>
                                </rdf:rest>
                            </rdf:List>
                        </rdf:rest>
                    </rdf:List>
                </owl:oneOf>
            </owl:DataRange>
        </rdfs:range>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:ID="manager">
        <rdfs:domain rdf:resource="#Department"/>
        <rdfs:range rdf:resource="#Employee"/>
    </owl:ObjectProperty>
</rdf:RDF>
```