DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé

Csongor Nyulas, Martin O'Connor, Samson Tu

Stanford Medical Informatics,
Stanford University School of Medicine, Stanford, CA 94305
csongor.nyulas<at>stanford.edu

Abstract: We present DataMaster, a Protégé plug-in that supports the importing of schema structure and data from relational databases into Protégé. The plug-in supports both OWL and frames-based ontologies and can be used with any relational database with JDBC/ODBC drivers.

Keywords: DataMaster, Protégé plug-in, Protégé OWL, tab widget, relational database, schema import, relational-to-OWL

1 Introduction

Importing data from relational databases into ontologies is frequently required, especially when an ontology is used to describe semantically the data used by a software application. Another growing category of applications needs database-ontology integration and/or interoperation, where a mapping between the database schema structure and ontology concepts is the main focus. In the latter cases the import of the data residing in relational databases may not be necessary or desired.

To meet these requirements, we have developed DataMaster, a Protégé plug-in that allows the user to import in a configurable way a relational database structure into a Protégé-OWL or Protégé-Frames ontology. The plug-in also supports the optional importing of table contents. The development of DataMaster was necessary, because existing plug-ins developed for importing data from relational databases into Protégé, such as DataGenie [1], do not support Protégé-OWL, schema-only import, and other import configurations available in DataMaster.

The DataMaster plug-in has been developed in the BioSTORM [2] project, which aims to develop a computational test bed that can draw on real-world data sources and that will allow users to configure, run, and evaluate alternative surveillance methods. The plug-in represents an important part of the semantic data-access layer, which annotates and integrates disparate data sources into a semantically uniform data stream.

In the following sections, we will describe different ways of importing a database structure and its content into an ontology using the DataMaster plug-in. Section 2 gives an overview of different ontologies for capturing the structure of a database, which are supported by different import configurations of DataMaster. Section 3 gives a short description of the DataMaster graphical user interface.

2 Ontologies for Describing Database Schemas

There are a variety of ways of describing the schema of a database in an ontology depending on the requirements of an application. For example, some applications will only require an import of the database content without needing a "live" connection to the database. In other cases, the mapping between the database structure and the ontology elements is more important, so that data reside in the database but are accessible to querying or reasoning through an ontology layer. We have designed four ontologies (three OWL ontologies and one Frames ontology) for describing database structures that are suitable for different types of applications and use cases.

2.1 Schema Structure Ontology for Protégé-Frames

DataMaster may be used to import a relational database structure and the table data into a Protégé-Frames ontology. The ontology for describing the database structure (Figure 1) is the same as the one used by the DataGenie plug-in. All imported database tables are defined as Protégé classes that are instances of the `Table Metaclass` meta-class. Each column of the database table is represented by a template slot added to the newly created table classes. The column slots will have data types corresponding to the SQL types associated to the database columns.

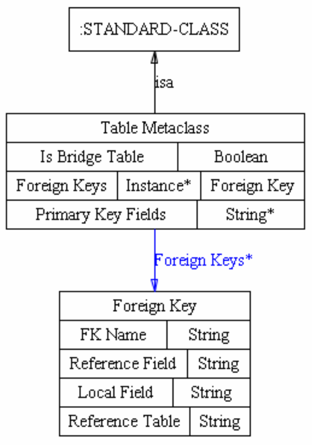


Figure 1. Database schema ontology for Protégé Frames

If there are foreign keys defined between the database tables, for each foreign key an instance of the `Foreign Key` class will be created that will be used to link the corresponding ontology classes.
It is also possible to import the data from the tables in the database: for each row in the table an instance of the table class will be created, and the values of the own slots at these instances will be set with values contained in the table row corresponding to the table columns. An extra slot of type instance will be created for each foreign key defined in the table that will point to the instances corresponding to the referred rows.

2.2 Schema Structure Ontology for Database Tables as OWL Classes

One of our goals when designing the schema structure ontology in OWL was to be able to use DL reasoning on it. This means that certain constructs used in the Frames approach (Section 2.1) had to be changed. However, certain combination of import options will result in an OWL Full schema ontology.
The Protégé OWL schema ontology for importing tables as classes is the OWL version of the previously described ontology for Protégé Frames. There are only a few differences:

- The imported classes will not be instance of a common meta-class, because it would make the ontology OWL Full. Instead, all the template slots attached to `Table Metaclass` in the Frames ontology, have been defined as annotation properties on the imported table classes.
- The property and class names from the Frames ontology are using the camel notation and the space character in the class and property names have been avoided. E.g., `hasForeignKeys` instead of `Foreign Keys`, `ForeignKey` instead of `Foreign Key`, etc.
- For easier navigation and performance reasons, we have introduced four additional object properties attached to the `ForeignKey` class to refer the local and referenced table classes as well as the referring and referred column properties.

2.3 Schema Structure Ontology for Database Tables as OWL Instances using Relational.OWL

Another way of representing the structure of a database in an OWL ontology is the approach taken by Relational.OWL [3]. The ontology used by Relational.OWL is shown in Figure 2. It defines four classes `Database`, `Table`, `Column` and `PrimaryKey`. The instances of these classes and their relationships can represent the schema structure of any relational database. One of the available configurations of DataMaster is to import the database structure as instances of the Relational.OWL ontology.
However, the Relational.OWL ontology is OWL Full, because Relational.OWL defines the representation of the database column types by using the `rdfs:range` property on the `Column` class, which makes the `Column` class be an `owl:Class` and an `rdf:Property` at the same time. We have provided in DataMaster two OWL DL modeling alternatives to the OWL Full constructs in Relational.OWL for the representation of the column datatypes. One modeling alternative is to attach a property `hasXSDType` to the Column class. The second alternative is to define a class `ColumnType` and a property `hasColumnType` that relates a `Column` instance to a `ColumnType` instance.

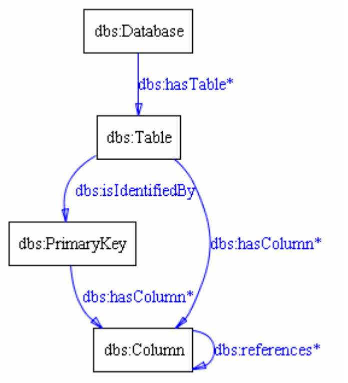


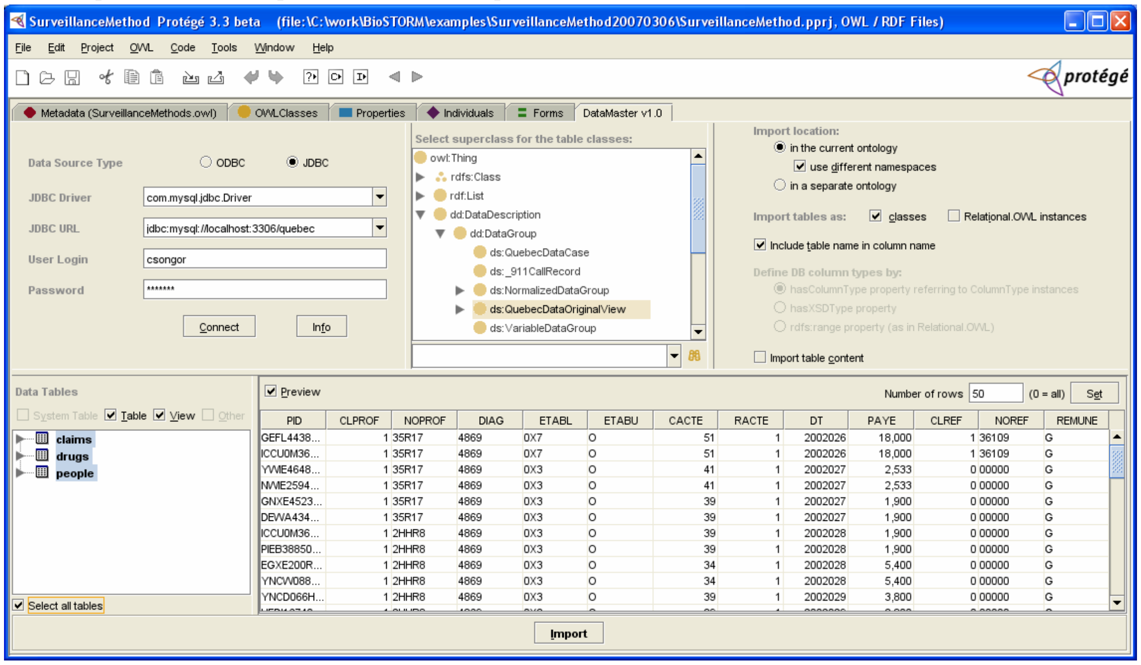
Figure 2. Relational.owl ontology

2.4 Schema Structure Ontology for Database Tables using meta-classes

In the Relational.OWL schema representation, the database tables are represented as OWL individuals of the `Table` class. This representation is very useful if we are only interested in importing the database structure in the ontology and not its data. This type of import is suitable for the case in which the data has to reside in the database but it can be accessed from an ontology layer by means of an OWL-database bridge.
However, there are cases in which it is desirable to import also the database content into the ontology. Relational.OWL provides this functionality by making the `Table` class a meta-class. Its instances corresponding to the imported tables will be OWL classes and the rows in the database tables will be instances of the table classes. In a similar way, the `Column` class will be a subclass of `owl:DatatypeProperty` and its instances corresponding to the database columns will be assigned to the row instances with values from the database.

3 The DataMaster user interface

A screenshot of the DataMaster plug-in is shown in Figure 3. The graphical interface consists of several sub-panels. In the upper left *connection panel* the user can select the database connection type (ODBC or JDBC), specify the JDBC or JDBC/ODBC driver, the data source name, the user name and the password to access the database. Pressing the *Connect* button will open a JDBC connection to the database and will activate the *table selection panel* and the *preview panel*, where the user can select the tables to be imported into Protégé and can get a preview of the selected tables. The *superclass selector panel* allows the user to select superclass(es) for the imported table classes. In this way, if the importing ontology already defines a class to represent database tables, the user can select this class in the superclass selector panel, and all the imported classes will be created as subclasses of that class.


SurveillanceMethod Protégé 3.3 beta (file:\C:\work\BioSTORM\examples\SurveillanceMethod20070306\SurveillanceMethod.pprj, OWL / RDF Files)
File
Edit
Project
OWL
Code
Tools
Window
Help
protégé
Metadata (SurveillanceMethods.owl)
OWLClasses
Properties
Individuals
Forms
DataMaster v1.0
Connect
Info
Set
Import

Data Source Type: ○ ODBC [✓ ● JDBC]

JDBC Driver: com.mysql.jdbc.Driver

JDBC URL: jdbc:mysql://localhost:3306/quebec

User Login: csongor

Password: *******

Select superclass for the table classes:

- owl:Thing
  - rdfs:Class
  - rdf:List
  - dd:DataDescription
    - dd:DataGroup
      - ds:QuebecDataCase
      - ds:_911CallRecord
      - ds:NormalizedDataGroup
      - ds:QuebecDataOriginalView
      - ds:VariableDataGroup

Import location:
[✓ ● in the current ontology]
[✓ use different namespaces]
○ in a separate ontology

Import tables as: [✓ classes] ☐ Relational.OWL instances

[✓ Include table name in column name]

Define DB column types by:
● hasColumnType property referring to ColumnType instances
○ hasXSDType property
○ rdfs:range property (as in Relational.OWL)

☐ Import table content

Data Tables

☐ System Table [✓ Table] [✓ View] ☐ Other

- claims
- drugs
- people

[✓ Select all tables]

[✓ Preview]

Number of rows 50 (0 = all)

PID	CLPROF	NOPROF	DIAG	ETABL	ETABU	CACTE	RACTE	DT	PAYE	CLREF	NOREF	REMUNE
GEFL4438...	1	35R17	4869	0X7	O	51	1	2002026	18,000	1	36109	G
ICCU0M36...	1	35R17	4869	0X7	O	51	1	2002026	18,000	1	36109	G
YWIE4648...	1	35R17	4869	0X3	O	41	1	2002027	2,533	0	00000	G
NWIE2594...	1	35R17	4869	0X3	O	41	1	2002027	2,533	0	00000	G
GNXE4523...	1	35R17	4869	0X3	O	39	1	2002027	1,900	0	00000	G
DEWA434...	1	35R17	4869	0X3	O	39	1	2002027	1,900	0	00000	G
ICCU0M36...	1	2HHR8	4869	0X3	O	39	1	2002028	1,900	0	00000	G
PIEB38850...	1	2HHR8	4869	0X3	O	39	1	2002028	1,900	0	00000	G
EGXE200R...	1	2HHR8	4869	0X3	O	34	1	2002028	5,400	0	00000	G
YNCW088...	1	2HHR8	4869	0X3	O	34	1	2002028	5,400	0	00000	G
YNCD066H...	1	2HHR8	4869	0X3	O	39	1	2002029	3,800	0	00000	G

Figure 3. Screenshot of the DataMaster plug-in

In the upper right panel, the user can specify some important additional import options:

- By selecting one of the "*Import location*" radio buttons and check boxes, the user can specify the physical location of the imported OWL classes and individuals. The import location can be the currently loaded ontology, or a separate ontology, which will be imported in the current ontology. If the user chooses to import the classes in the current ontology, he or she may choose to use a different namespaces for the imported classes. These options are currently not available for Protégé-Frames.
- The option "*Import tables as: classes and/or instances*", available in Protégé-OWL, allows the user to choose between one of the ontology representations of the database structure described in Section 2 that will be used in the import.
- The checkbox "*Include table name in column name*" is an option available when database tables are imported as classes. If selected, the column property names will be prefixed with the table name.
- If the table classes are to be imported as Relational.OWL instances, the data type of a column can be represented in one of the three ways outlined in Section 2.3.
- By selecting the "*Import table content*" checkbox the table data will be also imported. Note that importation of large databases may require increasing the Java VM heap size available for Protégé.

4 Conclusion

In this paper we have presented the DataMaster Protégé plug-in, which allows a user to import schema structures and data from relational databases accessible through JDBC. We have presented the four ontologies that can be used to represent the database structure and the table data and finally we have given a short overview of the different import options available in the DataMaster plug-in.

5 Acknowledgements

This work was supported by the Centers for Disease Control and Prevention under grant number SPO-34603. We would also like to acknowledge the DataGenie plug-in, parts of which were adopted by DataMaster.

References

[1] DataGenie wiki page: http://protege.cim3.net/cgi-bin/wiki.pl?DataGenie

[2] M. Crubézy, M. J. O'Connor, D. L. Buckeridge, Z. S. Pincus, M. A. Musen. Ontology-Centered Syndromic Surveillance for Bioterrorism. *IEEE Intelligent Systems*, 20(5):26-35. 2005

[3] C. P. de Laborda, S. Conrad. RelationalOWL - A Data and Schema Representation Format Based on OWL. *Conceptual Modelling 2005*, 43:89-96. 2005