Source URL: https://github.com/protegeproject/ontop-tutorial/wiki/Grocery-Tutorial

ontop-tutorial.md

This tutorial will give you a quick introduction of Ontop plugin. We are going to guide you step-by-step through the process of creating the mapping model and writing some queries against your domain model.

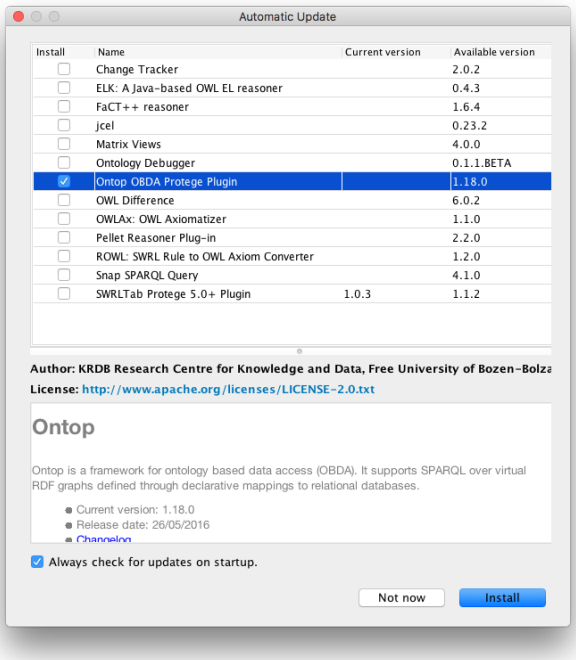
In this short tutorial you will:

- learn on how to work with the Ontop UI,
- understand Ontop mapping language to create some Relational-to-RDF mappings,
- make SPARQL queries and execute them to fetch the data.

The estimated completion time: 30-45 minutes.

Setup and Installation

The Ontop plugin is available through Protégé plugin auto-update, or by manually check at File > Check for plugins.... Select the "Ontop OBDA Protege Plugin" and proceed to the Install button. You might need to restart Protégé after the installation.



Next, you need to download several files:

- Standalone Grocery database (using H2) -- alternative URL: https://git.io/vPjyj
- Grocery ontology -- alternative URL: https://git.io/vPjSU
- (optional) Ontop mappings -- alternative URL: https://git.io/vPjS8
- (optional) Query examples -- alternative URL: https://git.io/vPjSB

(You might need to use "Save Link As..." in your browser if the download doesn't happen immediately)

Running H2 Database

Unzip the posc-grocery-h2.zip and navigate to its root directory using your favorite terminal application. Run the database using the given executable script.

In Linux or UNIX systems:

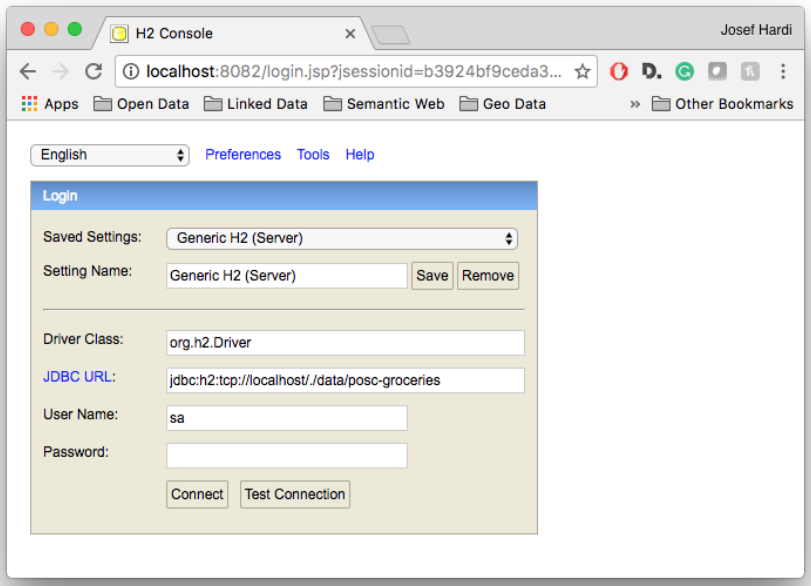
```
$ cd h2
$ ./h2.sh
```

In Windows:

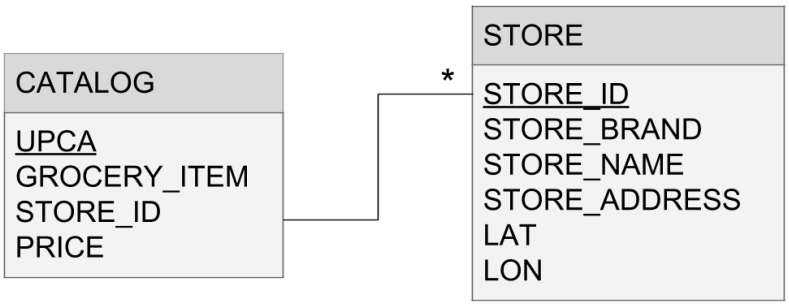
```
> cd h2
> h2.bat
```

A browser window will appear with a welcoming page to H2 database. Fill out the connection parameters and select Connect.

- Saved Settings: `Generic H2 (Server)`
- Driver Class: `org.h2.Driver`
- JDBC URL: `jdbc:h2:tcp://localhost/./data/posc-groceries`
- User Name: `sa`
- Password: (leave blank)



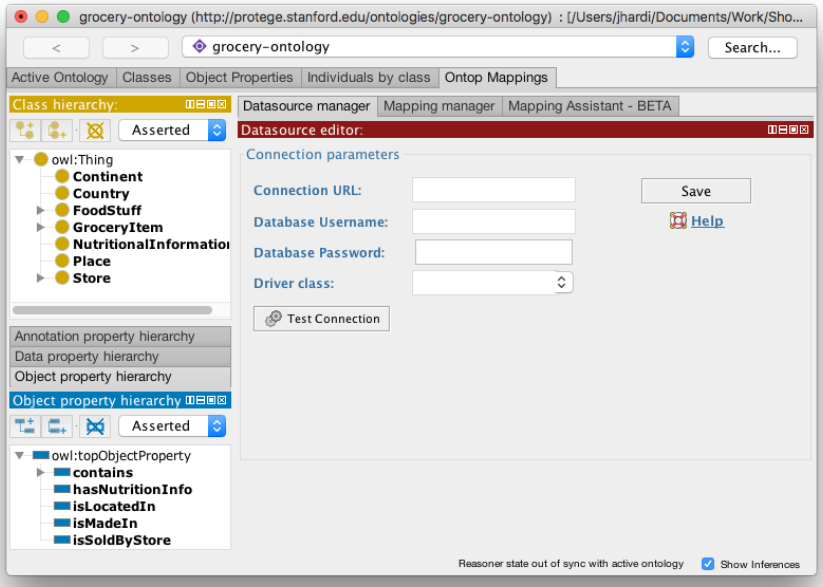
You should be able to access our simple database and check its two tables: "CATALOG" and "STORE" and the relationship.



STEP 1 - Starting Ontop

Open the grocery ontology using Protégé and make sure it is currently the active ontology.

To open the Ontop tab, select Window > Tabs > Ontop Mappings.



STEP 2 - Installing H2 Driver

Go to Preferences > JDBC Drivers and select Add. Type the input parameters as below:

- Description: `H2 Driver`
- Class name: `org.h2.Driver`
- Driver File (jar): (Select Browse and point to our previously H2 root directory and choose h2-1.4.191.jar)

Select OK to finish and you should see the "H2 Driver" in the list with status "Ready".

Select OK to close the Preferences dialog.

STEP 3 - Establishing the Database Connection

Return to the "Ontop Mappings" tab and make sure the sub-tab "Datasource manager" is selected. Fill out the connection parameters. You might be noticing that the values are similar to our previous H2 database setup.

- Connection URL: `jdbc:h2:tcp://localhost/./data/posc-groceries`
- Database Username: `sa`
- Database Password: (leave blank)
- Driver class: `org.h2.Driver`

Select the Test Connection and make sure you get a green message saying: "Connection is OK"

STEP 4 - Creating Class Assertion Mapping

In this step, we are going to create a class assertion mapping. First, we need to go to the sub-tab "Mapping manager" to create a mapping model. Ontop requires this model so it can understand the connection between the underlying data and the ontology.

To create a new mapping, select the Create button and fill out the parameters as below and select Accept to save.

- Mapping ID: `Walkers Shortbread`
- Target (Triples Template):

```
:{UPCA} a :WalkersShortbreadItem
```

- Source (SQL Query):

```
select * from CATALOG where GROCERY_ITEM = 'Walkers Shortbread'
```

> Note: It is better to know first your data. Typing the SQL query first and run Test SQL Query will give you the overview about the column names and the data values. Our SQL query will return you four columns, i.e., "UPCA", "GROCERY_ITEM", "STORE_ID", and "PRICE".
>
> Note: The `:{UPCA}` is called the column reference notation. The colon `:` character indicates the default prefix label. Later when Ontop processes this notation, it will generate an IRI based on the default prefix and the data from "UPCA" column.
>
> Note: The `a` is a short form of `rdf:type` used for class assertion.
>
> Note: Ontop requires all entity names be present in the input ontology. Otherwise, it will complain as value missing and you won't be able to save the mapping.

STEP 5 - Creating Data Property Assertion Mapping

In this step, we are going to add some data property assertions to our previous mapping. Do a double-click in the mapping item and update it into:

```
:{UPCA} a :WalkersShortbreadItem ; :productName {GROCERY_ITEM} ; :price {PRICE}
```

> Note: The `:productName` and `:price` are the data properties in our Grocery ontology. The `{GROCERY_ITEM}` and `{PRICE}` are the column references. Notice no prefix label is used in these references such that Ontop will generate a typed literal value instead of an IRI.
>
> Note: Use a semi-colon `;` character to separate between different mappings.

STEP 6 - Creating Object Property Assertion Mapping

In this step, we are going to add an object property assertion to the same mapping. Do a double-click in the mapping item and update it into:

```
:{UPCA} a :WalkersShortbreadItem ; :productName {GROCERY_ITEM} ; :price {PRICE} ; :isSoldByStore :{STORE_ID} .
```

> Note: The `:isSoldByStore` is the object property in our Grocery ontology. The `:{STORE_ID}` is the column reference. Notice the prefix label `:` in the reference.
>
> Note: You might now see the pattern in the mapping language: 1) entity names are always accompanied by a prefix label, 2) the value for data property is always a typed literal so no prefix label in the column reference and 3) in contrast, the value for object property is always an IRI so the column reference requires a prefix label.
>
> Note: Use a period sign `.` to end the mapping.

STEP 7 - Write Queries to Test the Mappings

Open the Query tab by selecting Window > Tabs > Ontop SPARQL.

Type this sample query below and select Execute to run the query execution.

```
PREFIX : <http://protege.stanford.edu/ontologies/groceries/>
SELECT * WHERE {
   ?x a :WalkersShortbreadItem.
}
```

> Note: If you get an error message saying "Quest must be started before using this feature" then it means you need to activate the reasoner first. Go to Reasoner > Ontop 1.18.0 and then select Reasoner > Start reasoner.

You should get these following results:

```
<http://protege.stanford.edu/ontologies/groceries/809-74-4400>
<http://protege.stanford.edu/ontologies/groceries/224-19-6622>
```

```
<http://protege.stanford.edu/ontologies/groceries/921-84-3998>
<http://protege.stanford.edu/ontologies/groceries/712-26-7903>
<http://protege.stanford.edu/ontologies/groceries/597-94-5307>
<http://protege.stanford.edu/ontologies/groceries/997-95-7786>
<http://protege.stanford.edu/ontologies/groceries/451-67-4590>
```

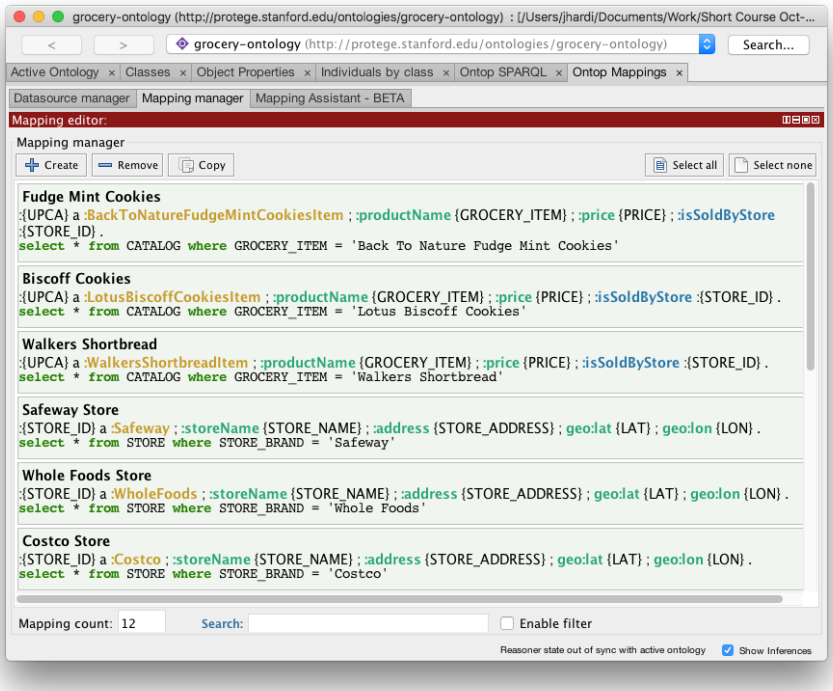
> Note: The returned string are the values for `?x` variable in the query. They are IRIs generated from the column reference notation `:{UPCA}`.

Let's ask for some properties in the query and look at the returned results.

```
PREFIX : <http://protege.stanford.edu/ontologies/groceries/>
SELECT * WHERE {
   ?x a :WalkersShortbreadItem;
      :productName ?productName;
      :price ?price;
      :isSoldByStore ?store .
}
```

STEP 8 - Make More Mappings

At this point, you might get an intuition about creating mappings in Ontop. To speed up things around, reopen the Grocery ontology but this time make sure the Ontop mappings (posc-groceries.obda) and Query examples (posc-groceries.q) files are stored in the same location as the ontology file. Ontop will automatically load these files and you should see more mappings in the "Mapping manager" tab (i.e., a total of 12 mappings).



Feel free to have a pause here to learn those new mappings.

STEP 9 - Make More Queries

Go to "Ontop SPARQL" tab and you should see 5 example queries. Run each one of them and feel free to study each query.

> Note: Look at "Example-03" query where it asks to show all grocery items. Notice that our mappings do not have a class declaration about GroceryItem. But our ontology says that all product items (e.g., Walker's Shortbread, Fudge Mint Cookies, and so on) are subclasses of GroceryItem and Ontop incorporates this knowledge to answer the query precisely. This is one of the key features that Ontop uses a reasoner for answering queries.

Conclusion

Ontop provides a non-intrusive solution for businesses that would like to add a semantic layer into their existing data storage. The solution is more cost-effective for the early adopters in triplestore with better knowledge in SQL databases. The concept of ontology-based data access that Ontop is bringing is one of the best solutions for data integration.

Sample Prototype

The EU Optique project is one big user that uses Ontop (or ontology-based data access in general) to create a semantic data integration platform. You can access the demo here: http://optique-northwind.fluidops.net/ and login with username `demo` and password `demo`.

The demo application has a very limited access but you could try their Visual SPARQL tool at "VQS" menu item. Use the default settings and load the interface.

Help and Support

Ontop is an open source project managed by the Knowledge Representation and Database (KRDB Research Centre) in Italy.

If you have questions about Ontop and Ontology-based Data Access, please address to https://groups.google.com/d/forum/ontop4obda

Official website: http://ontop.inf.unibz.it/