TopQuadrant™

Using TopBraid to Work with Spreadsheet Data

By Irene Polikoff, CEO, TopQuadrant, Inc.

Last Updated: March 24, 2015

Overview

Much data today is held in spreadsheets. Not surprisingly, there is a lot interest in transforming spreadsheet data into RDF.

Spreadsheets are very flexible. The most commonly used format is a simple table with a column header on top. But, there are spreadsheets that communicate hierarchical structures, which contain multiple tables within a single worksheet and have other, potentially complex, semantics.

To address the need to capture this type of information and the semantic relationships encoded in the spreadsheet structure, the TopBraid product family offers different capabilities for transforming spreadsheets into the semantic standards representation RDF. These capabilities range from simple "out of the box" mappings to more sophisticated transformations.

This technical whitepaper provides a description of the set of TopBraid capabilities for importing and transforming different types of spreadsheet data into meaningful, structured information in RDF, along with guidance for when and how to use each specific capability.

Tables 1 and 2 below work together to provide a listing of each of the TopBraid capabilities (or options) for importing spreadsheets depending their structure, and decision criteria for when to select which option to use. These tables are followed by an extended section that provides detail on how and when to use each of option.

Table 1: Decision criteria for deciding which TopBraid capability to use.

1. Is your spreadsheet a table with column headings in the first row?	*If yes, use option 1, 2 or 4. Also, see questions 4 through 6.*
2. Does your spreadsheet contain a hierarchy?	*If yes, use option 5.*
3. Does your spreadsheet contain arbitrary complex data structures?	*If yes, use option 3.*
4. Does the tabular data in your spreadsheet contain relationships among cells in a given row? That is, does each row contain information about multiple interrelated entities?	*If yes, use option 4.*
5. Do you need to provide a web user interface to users	*If yes, use option 4 or 5.*

who will be importing spreadsheet data?	
6. Do you need to be able to incrementally upload changes (deletions as well as additions) in the data after your initial import?	*If yes, options 4 or 5 will provide this feature "out of the box." With other options, you would need to create scripts.*

Table 2: List of TopBraid capabilities (options 1-5) for importing spreadsheets.

Capability	Tool Providing User Interface	SPARQLMotion Module or Servlet for Automation	Spreadsheet Structure Assumptions
1. Auto-conversion on the fly using "semantic tables"	TopBraid Composer	sml:ImportRDFFromWorkspace	One table per worksheet
2. Import of tabular spreadsheets that are saved in a tab-separated format	TopBraid Composer	sml:ConvertSpreadsheetToRDF	One table
3. Import of arbitrary spreadsheets using a "spreadsheet ontology"	TopBraid Composer	sml:ImportExcelCellInstances	None
4. Import of tabular spreadsheets using TopBraid EVN or TopBraid RDM	Web UI in TopBraid EVN* and TopBraid RDM*	importFileUpload servlet	One table per worksheet; will import one worksheet at a time
5. Import of hierarchical spreadsheets using TopBraid EVN or TopBraid RDM	Web UI in TopBraid EVN* and TopBraid RDM*	importFileUpload servlet	One hierarchy per worksheet, will import one worksheet at a time

**Note: These capabilities can also be used from within TopBraid Composer Maestro Edition since it includes demo versions of the respective server products.*

How and when to use each "import" capability

The following sections provide more information on how and when to use each of options 1-5 for importing spreadsheets and transforming their content into RDF.

Each section includes the following information:

- How to invoke use of the specific capability through a UI
- Advantages and limitations of the option
- An optional worked example, results and other details
- How to use the capability through an automated script (if possible)

1. Auto-conversion on the fly using "semantic tables"

To use this capability, within TopBraid Composer (TBC) double click on a spreadsheet file and the auto-conversion to RDF will happen on the fly.

Supported Formats: .csv, .tsv, .xls, .xlsx
Advantages:
• No mappings to do, everything is generated automatically. • As an elaboration of this option it is also possible to use an existing ontology and map it to spreadsheet data using annotations. • Roundtrip-able: changes made in TopBraid Composer are saved back in the spreadsheet. • Can convert multiple worksheets per spreadsheet. • Directly uses Excel files • Can be invoked as part of a SPARQLMotion or SPARQL Web Pages (SWP) web service or script.
Limitations
• Each worksheet must contain a single table. • Creates one resource per row. Only creates datatype properties. If a row represents a resource and its relationships to other resources, the user will need to do additional transformations after the conversion. • There is no control over URI creation. • No web user interface is provided for this option; a user must use it within TBC or utilize the capability through a web service/script.

For each worksheet, the auto-conversion will create a class, then create properties corresponding to the column headings (the first row), and, finally, will create a resource (as a member of the "worksheet class") for each of the rows below the first one. It will give these resources property values based on the cells in the row.

As an illustrative example, consider the tabular spreadsheet file with contacts shown below:

First Name	Last Name	E-mail	Phone	Title	Company	Company Size
Tom	Jones	tjones@acme.com	111-222-3333	Manager	ACME Corporation	Large
Jane	Doe	jdoe@example.com	444-555-6666	Vice	Example	Medium

Assuming that the file name of this spreadsheet is Workbook1.xlsx, the RDF representation of the first row will look as follows (properties are listed in alphabetic order).

```
Workbook1:Row-0  a              Workbook1:Sheet1 ;
        Workbook1:company       "ACME Corporation"^^xsd:string ;
        Workbook1:companySize   "Large"^^xsd:string ;
        Workbook1:eMail         "mailto:tjones@acme.com"^^xsd:anyURI ;
        Workbook1:firstName     "Tom"^^xsd:string ;
        Workbook1:lastName      "Jones"^^xsd:string ;
        Workbook1:phone         "111-222-3333"^^xsd:string ;
        Workbook1:title         "Manager"^^xsd:string ;
        tables:rowIndex         "0"^^xsd:int .
```

Each of the properties will have a declaration that includes a column index property, as shown in the following for the First Name property.

```
Workbook1:firstName  a       owl:DatatypeProperty ;
        rdfs:domain          Workbook1:Sheet1 ;
        rdfs:label           "First Name" ;
        rdfs:range           xsd:string ;
        tables:columnIndex   "0"^^xsd:int .
```

URIs for resources will be derived using the file's location, file name and row number. If the spreadsheet is located directly in the example.com project in the TopBraid workspace, the full URI for the resource represented by the `Workbook1:Row-0` row will be `file:///example.com/Workbook1.xlsx#Row-0`.

A unique feature of this approach is that a user can make changes to the dynamically created RDF and save the result back to the spreadsheet. The RDF graph is never persisted – that is, no RDF file is created. If you need to create an RDF file, use Export -> Merge/Export RDF graphs.

You can also use an existing ontology and annotate it to let TopBraid know what classes and properties to use:

- Import table.ttl (from the TopBraid project under the TBC folder) into your ontology, add the property tables:sheetIndex to the class you want to populate with the spreadsheet data and give it value 0 for the first worksheet.
- Then, make sure that each column property corresponding to the worksheet data has this class as its rdfs:domain.
- Give each column property a tables:colIndex value, starting with 0.
- Repeat with the classes for the additional worksheets as required.
- When you double click on the file to open it, indicate that you have an existing ontology to use.

This capability can also invoked within automated scripts. The `sml:ImportRDFFromWorkspace` module and any other module that loads RDF will accept a spreadsheet file as its input (as if it was already in RDF) performing the conversion described here.

2. Import of tabular spreadsheets that are saved in a tab-separated format

To use this capability, within TopBraid Composer (TBC) right click on a tab-delimited spreadsheet file and select the Import menu option. Then, in the Select dialog box under the choice 'TopBraid Composer', select: Import Tab-Delimited Spreadsheet file. From there, a UI wizard will lead you through the conversion process.

Supported Formats: typically .txt
Advantages:
• No mappings to do; everything is generated automatically. • As an elaboration of this option it is also possible to use an existing ontology and map it to spreadsheet data using annotations. • A user-friendly wizard is provided for mapping columns to the properties in the ontology. • Control over URI creation is provided. • Can be invoked as part of a SPARQLMotion or SPARQL Web Pages (SWP) web service or script.
Limitations
• Each worksheet must become a separate file and be converted separately. • Each file must contain a single table. • The conversion creates one resource per row, and only creates datatype properties. If a row represents a resource and its relationships to other resources, the user will need to do additional transformations after the conversion. • No web user interface is provided for this option; a user must use it within TBC or utilize the capability through a web service/script.

The first step in the wizard is to provide a base namespace for the RDF graph to be created. The base namespace will also be used to form the URIs of imported resources. If there is no pre-existing ontology, then the entries in the first row will be treated as property names. The logic used for the conversion is very similar to option 1 above. To support working with existing ontologies, the wizard includes a "property mapping" screen.

This capability can also invoked within automated scripts. The `sml:ConvertSpreadsheetToRDF` module accepts a tab-delimited spreadsheet file and a set of parameters as its input, performing the conversion described here.

3. Import of arbitrary spreadsheets using a "spreadsheet ontology"

To use this capability, within TopBraid Composer (TBC) right click on a tab-delimited spreadsheet file and select the Import menu option. Then, in the Select dialog box under the

choice 'TopBraid Composer', select Import Excel File into Spreadsheet Ontology. You will be asked to provide a name for the converted RDF file and the base URI.

Supported Formats: .csv, .tsv, .xls, .xlsx
Advantages:
• No mappings to do, everything is generated automatically. • This option is the most flexible and granular of all importers — no assumptions are made about the structure of the spreadsheet or the meaning of its data; a resource is created for any cell that has data. • Can convert multiple worksheets per spreadsheet. • Can be invoked as part of a SPARQLMotion or SPARQL Web Pages (SWP) web service or script.
Limitations
• The output is rarely useful "as-is"— that is, additional transformations are needed to create a meaningful RDF graph. • No web user interface is provided for this option; a user must use it within TBC or utilize the capability through a web service/script

A spreadsheet ontology (spreadsheets.rdf, or spreadsheets.ttl before release 4.4) is available in TopBraid project under TBC folder. It is a simple model of a spreadsheet that uses three classes: ss:Workbook, ss:Sheet and ss:Cell.

An illustrative example of a complex spreadsheet that may be converted using this approach is the expense report shown below.

Example Enterprises
100 Main Street
Big City

1

Employee Name: _____ Purpose of Trip: _____
Client: _____
Location: _____

Dates From: _____ To: _____

TOTAL EXPENSES	
TOTAL TO BE REIMBURSED TO EMPLOYEE	

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat	Totals	Paid by Corporate Credit Card
Date:									
Auto Miles	0	0	0	0	0	0	0		
Amount	$ -	$ -	$ -	$ -	$ -	$ -	$ -	$ -	
Tolls/Parking								$ -	
Air Travel								$ -	
Hotel/Lodging								$ -	
Hotel Tax								$ -	
Telephone								$ -	
Tips								$ -	
Breakfast								$ -	
Lunch								$ -	
Dinner								$ -	
Car Rental								$ -	
Gasoline								$ -	
Taxi/Etc.								$ -	
Taxi/Etc.								$ -	
Laundry								$ -	
Bus. Meals								$ -	
Entertainment								$ -	
Other A.								$ -	
Totals	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$ -	$ -
Total to be Reimbursed = Expenses less amount paid by corporate credit card								$	-

The conversion creates one instance of a Workbook class, as many instances of the Sheet class as there are worksheets in the spreadsheet and an instance of a Cell class for each cell that contains data.

The workbook instance will point to the worksheets, worksheets will point to their cells. Each resource representing a cell has a row index, a column index, an ss:cellContents property which holds the data and an ss:cellType property that, depending on the cell data, will be set to values such as xsd:string.

Typically, after using this method to convert spreadsheet data to RDF, users run SPARQL queries, often implemented as a set of SPIN rules, to transform it into meaningful RDF.

This capability can also be invoked within automated scripts using the `sml:ImportExcelCellInstances` module.

4. Import of tabular spreadsheets using TopBraid EVN or TopBraid RDM

To use this capability, within TopBraid EVN or TopBraid RDM select a vocabulary you need to import spreadsheet data into. Then, go to the Import tab and select: Import Spreadsheet Using Pattern. From there, a UI wizard will lead you through the conversion process.

Supported Formats: .csv, .tsv, .xls, .xlsx
Advantages:

• A user-friendly web wizard for mapping columns to the properties in the ontology. • Control over URI creation is provided. • Can create relationships from the data. • Can be automated by saving the mapping template. • Provides an "overwrite" option that will overwrite existing data with updated information. • Can log a history of changes.
Limitations
• A file may contain multiple worksheets, but each import only deals with one worksheet at a time (selected in the wizard). • Requires the worksheet to have a tabular structure.

The first step in the wizard is to select the spreadsheet to import. You will also select a worksheet index and a class for which the data is being imported. The second step is to select the spreadsheet pattern. For this conversion option, select "No Hierarchy." The final step is about mapping columns to properties. The first row of the spreadsheet is expected to contain column names.

On the mapping page you will also be able to select an "overwrite" option. When selected, TopBraid will compare data in the spreadsheet with the data you already have in the dataset and will adjust it accordingly. For example, if you are importing a spreadsheet with a list of contacts and the spreadsheet version imported first had the phone number for one of the contacts as 111-222-3333 and the second version had it as 999-222-3333, this option will delete 111-222-3333 and add 999-222-3333.

Columns can be mapped to attributes using datatype properties and relationships using object type properties. A column mapped to a relationship must contain values of the primary key property for the range class.

A primary key is a property whose values are used to generate resource URIs in EVN or RDM. There can be only one primary key property for a class. The value of this property must be unique among all the members of a class – for example, an employee ID or a social security number for people, or the ISO alpha-2 character code for countries. In the absence of a primary key, resource URIs are generated from labels by using a custom plug-in mechanism that is installation-specific, or entered by users when they manually create a new resource.

If you are importing a spreadsheet with the list of contacts in the example shown earlier under option 1, the "Company" column can be used to create a relationship between a person and the company they are associated with. To do this, note the following requirements and steps:

- The class at the range of the relationship (in this case, the class Company) must have a primary key defined.
- The column mapped to a relationship must contain values corresponding to the primary key.

  - It could contain a company name, as in the tabular spreadsheet example shown earlier, if "name" is the primary key for the Company class.
  - If the primary key is some other identifier (for example, a stock symbol or DUNS number), the column should contain stock symbols or DUNS numbers, as appropriate.
- Import the spreadsheet for the Person class and map the "Company" column to the relationship you are creating between Person and Company (for example, "works for"). The import process will take the data in the "Company" column and turn it into a URI of the related resource. This will create a relationship, but it will not identify the related resource as a company.
- Import the spreadsheet again if you need to create instances of the Company class. That is, capture the fact that related resources are companies and/or import any information about companies themselves.
  - For example, "Company Size" information (shown in the tabular spreadsheet above) is clearly information about companies, not people.
  - This time, select Company as the class to import data for. Ignore columns that contain information about people.

The mappings that are created using the wizard can be saved as a template. The next time a spreadsheet with the same structure needs to be imported, to save time, you can use "Import Spreadsheet Using a Template." This is another spreadsheet processing option available on the Import tab that extends some of the options covered in this paper. It requires the definition of a template first. Templates contain data transformation rules. In addition to creating templates by saving mappings used to do an import, for more complex transformations, templates can be created in TopBraid Composer using SPINMap (see more on SPINMap in the Summary section below).

5. Import of hierarchical spreadsheets using TopBraid EVN or TopBraid RDM

To use this capability within TopBraid EVN or TopBraid RDM select a vocabulary you need to import spreadsheet data into. Then, go to the Import tab and select Import Spreadsheet Using Pattern. From there, a UI wizard will lead you through the conversion process.

Supported Formats: .csv, .tsv, .xls, .xlsx
Advantages: • A user-friendly web wizard for mapping columns to the properties in the ontology. • Control over URI creation is provided. • Can create relationships from the data. • Can be automated by saving the mapping template. • Provides an "overwrite" option that will overwrite existing data with updated information. • Can log a history of changes.
Limitations

- A file may contain multiple worksheets, but each import only deals with one worksheet at a time (selected in the wizard).
- Requires the worksheet to have a hierarchical structure that is represented using one of the supported hierarchical patterns.

The first step in the wizard is to select the spreadsheet to import. You will also select a worksheet index and a class for which the data is being imported. The second step is to select the spreadsheet pattern. Select a pattern that aligns with how the hierarchy is represented in your spreadsheet. The final step involves mapping columns to properties. The first row of the spreadsheet is expected to contain column names.

On the mapping page you will also be able to select an "overwrite" option. When selected, TopBraid will compare data in the spreadsheet with the data you already have in the dataset and will adjust it accordingly.

As noted for option 4, the next time a spreadsheet with the same structure needs to be imported, to save time, you can also use "Import Spreadsheet Using a Template."

In Summary

There are many ways that people use spreadsheets. TopBraid provides an extensive set of capabilities as options for transforming spreadsheet data into RDF. You can learn more about capabilities 1 through 3 by reading the Help files in TopBraid Composer. Capabilities 4 and 5 are described in more detail in the user guides for TopBraid EVN and TopBraid RDM.

Once your data is in RDF, SPINMap provides a good approach for defining additional transformations where needed. SPINMap is a TopBraid technology that uses a visual programming approach for mapping one model to another using SPIN functions to perform model transformations. See an example of using SPINMap for more insight on the use of this approach. SPINMaps can be used as templates by the EVN and RDM "Import using a Template" feature.

Just as we want to bring spreadsheet data into RDF, we also often want to create spreadsheets from RDF. Each TopBraid product provides a way to save the results of queries and searches as spreadsheets. Using option 1 (semantic tables), one can edit spreadsheet data directly in TopBraid Composer. In addition to spreadsheet import modules, there are also spreadsheet export modules.

We hope that this white paper provided you with enough information to easily bring any spreadsheet into the world of RDF and Linked Data. If you have more questions about the capabilities described here or about TopBraid products in general, we invite you to write to us at info@topquadrant.com.

About the Author

Irene Polikoff is a Co-Founder and CEO of TopQuadrant, Inc. She has more than two decades of experience in software development, management, consulting and strategic planning. Since co-founding TopQuadrant in 2001 Irene has been involved in more than a dozen projects in government and commercial sectors. She has written strategy papers, trained customers on the use of the Semantic Web standards, developed ontology models, designed solution architectures and defined deployment processes and guidance.

With the introduction of TopQuadrant's Semantic Solution Platform, TopBraid Suite, Irene has been leading the evolution of the company from a consulting firm to a software vendor. As an executive leader of TopQuadrant Irene is responsible for daily operations and long term strategy of the company.

Before starting TopQuadrant Irene was a Principal in the national Knowledge, Content Management and Portals Practice in IBM Global Services. Prior to that she was a Senior Development Manager and a Project Executive for IBM worldwide consultant's tools and methods.

Prior to IBM, Ms. Polikoff held IT management positions at Fortune 500 companies where she was responsible for development and deployment of enterprise-wide mission critical information systems. Irene has a Master's degree in Operations Research from Columbia University. She is co-author of the book Capability Cases: A Solution Envisioning Approach on software requirements and architecture

About TopQuadrant

TopQuadrant's standards-based solutions enable organizations to evolve their information infrastructure into a semantic ecosystem, the foundation for intelligent business capabilities and integrated big data. As a result, data can be organized, shared and exchanged regardless of its structure, origin or location. TopBraid Enterprise Vocabulary Net™ supports collaborative management of enterprise metadata, models, business glossaries and taxonomies used in search, content navigation and data integration. TopBraid Reference Data Manager™ supports the governance and provisioning of reference data, including the enrichment of reference datasets (code lists) with comprehensive metadata. TopBraid Insight™ is a semantic virtual data warehouse that enables federated querying of data across diverse data sources as if they were in one place. TopQuadrant customers include many government agencies and Fortune 1000 companies in numerous industries including pharmaceutical, financial services, energy and digital media. For more information, visit http://www.topquadrant.com