Land & Water

Summer Internship 2015/16



Developing Environmental Vocabularies

By Xavier Butcher – 12/01/16



Contents

[1 Keywords 2](#_Toc440266767)

[2 Introduction 3](#_Toc440266768)

[3 Example 3](#_Toc440266769)

[4 Materials 4](#_Toc440266770)

[5 Method 4](#_Toc440266771)

[5.1 Summarised Method 5](#_Toc440266772)

[5.2 Full Method 5](#_Toc440266773)

[5.2.1 Background Reading 5](#_Toc440266774)

[5.2.2 Database 5](#_Toc440266775)

[5.2.3 Excel (CSV) 8](#_Toc440266776)

[5.2.4 RDF123 12](#_Toc440266777)

[5.2.5 Text Editor (notepad++) 15](#_Toc440266778)

[5.2.6 TopBraid Composer 18](#_Toc440266779)

[5.2.7 Post Processing (Text) 22](#_Toc440266780)

[5.2.8 Publishing 24](#_Toc440266781)

[6 Results 27](#_Toc440266782)

[7 Discussion 27](#_Toc440266783)

[7.1 Key Issues 27](#_Toc440266784)

[7.2 Specific Issues – Lab Methods 28](#_Toc440266785)

[7.3 Future Developments 29](#_Toc440266786)

[8 Conclusion and Personal Reflection 30](#_Toc440266787)

[9 Acknowledgements 30](#_Toc440266788)

# Keywords

Concept, CSIRO, Database, Editors, Environment, Excel, Internship, Methods, Notepad, RDF, Registry, Semantics, SKOS, Soil, SPARQL, SQL, Taxonomy, TopBraid, URI, Vocabulary, Wiki.

# Introduction

The aim for this project was to create and collate a number of online vocabularies into a standard format and location. Vocabularies are often hosted online, and are used to support environmental applications. They contain labels, definitions and other related information for a particular concept or registry. This project mostly focussed on soil and land operations. Currently, a number of vocabularies are unique to their individual application, and are often in different formats (text, csv, pdf, or lists). Having a definitive vocabulary on a subject is critical, as the same term could have different definitions or meanings in different vocabularies.

It is also necessary to introduce semantic relationships, which essentially add another layer of detail to vocabulary. This means different terms can be placed in a hierarchical order (broader and narrower), as well as relating similar terms. These vocabularies follow the guidelines of SKOS, the Simple Knowledge Organization System, which is an industry standard of rules and guidelines across online vocabularies which aims to make publication and use of vocabularies an easy and standard process.

SPARQL the SPARQL Protocol and RDF Query Language is, as the acronym suggests, an RDF query language. It is tending to be the industry standard query language. SPARQL allows the user to query semantic data in databases, and retrieve and manipulate RDF data.

There were 4 main tasks relating to this project, which were completed successfully:

•          Source relevant existing vocabularies

•          Convert the vocabulary content using semantic web technologies (RDF/SKOS)

• Harmonising vocabulary content with existing vocabularies where applicable

•          Publish vocabularies to SPARQL triple stores where applicable

# Example

For those who have little or no knowledge of vocabularies, SKOS or semantics, this section will give a basic example of how these theories are used.

This example will centre on the ‘Concept’ of a “Margherita Pizza”. As a bare minimum, a Concept must have a Label (Margherita Pizza), a URI (Unique Resource Identifier – a unique web address where the concept can be located) and a Definition (Margherita Pizza is a flatbread generally topped with tomato sauce and cheese and baked in an oven). A Concept may also be part of a hierarchy of other Concepts. In this case, “Margherita Pizza” is a Narrower Concept of “Pizza”, which itself is a Narrower Concept of “Food”. A Concept can also be related to other concepts regardless of hierarchy, for instance “Margherita Pizza” is related to “Mushroom Pizza”. A Concept can also be part of a collection. For example “Margherita Pizza” can belong to the “Vegetarian Pizza” collection.

# Materials

* National Committee on Soil and Terrain (2009), ‘Australian soil and land survey field handbook (3rd edn).’ (CSIRO Publishing: Melbourne)
* ‘Soil Information Transfer and Evaluation System (SITES) – Database design and exchange protocols (version 2.0) (2012).’ Jacquier, D; Wilson, P; Griffin, T; Brough, D. (CSIRO Publishing: Canberra)
* ‘Soil Chemical Methods – Australasia’ (2011). Rayment, G; Lyons, D. (CSIRO Publishing: Melbourne).
* The CSIRO environmental informatics Wiki - <https://wiki.csiro.au/display/EI/Environmental+Informatics+Home>
* The CSIRO vocabularies and vocabulary services Wiki - <https://wiki.csiro.au/display/VOCAB/Home>
* The Linked Data Registry - <http://registry.it.csiro.au/sandbox/student/xavier>
* Microsoft office suite
* Notepad++
* TopBraid Composer
* RDF123
* Microsoft SQL Server Management Studio

# Method

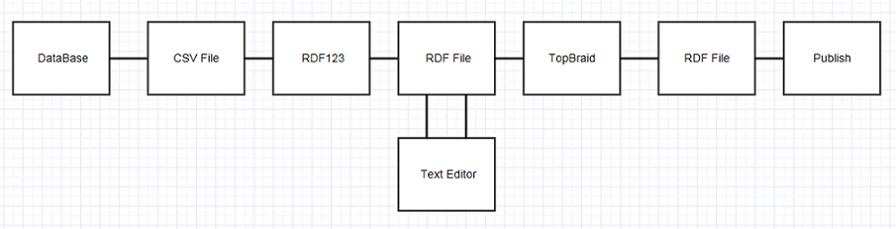
This report will explain in depth the method used to create a vocabulary. This can be seen in Figure 1. There are different programs that achieve the same outcome, however I recommend using the same programs Megan and I used, as you will be able to follow our notes. What I felt was lacking was a clear step by step method with screenshots and annotations, which is what this method intends to fix. A clearer version of this can be found on my wiki.

Figure : Process diagram

## Summarised Method

The first thing step is to do a lot of background reading. There are a number of documents, papers, manuals and wiki’s online that are very useful at getting you up to speed on vocabularies. Next, the soil related data you need can be found in the CSIRO natsoil database. This is viewed on Microsoft SQL Server Studio. Here, a collection of tables are kept which you can export as a CSV. Following this, you can open the CSV in Excel. This is where you will need to create and modify columns that will capture all of the properties you wish to maintain in the vocabulary. The most common manipulations will be the creation of a URI and the creation of a Label/Definition. Subsequently, you can import this CSV file into RDF123. This program essentially converts your tabulated data into an RDF format. This is also where many of the links between the data are created. After a quick tidy up in Notepad, you can import the file into TopBraid. TopBraid is a program that lets you create semantic relationships and add further necessary properties to make the vocabulary function better. Finally, after another tidy up in Notepad, you are ready to publish to the registry.

## Full Method

Section 5.2 gives an in depth look at a step by step method for creating a vocabulary.

### Background Reading

Prior to starting the creation of a vocabulary, it was necessary to undertake an intensive few days of background reading. I had little-to-no knowledge of informatics before starting this internship. Except for Excel, I had not used any of the other programs necessary for this work, so I had to learn these from scratch. I also had minimal knowledge of the actual concept of vocabularies, especially including SKOS, SPARQL, RDF or Semantics, so much of my time was spent getting up to speed on these concepts.

There are a number of Wiki’s, both from previous interns who had worked in this area previously, as well as those managed by the Environmental Informatics team. I found that the most useful notes are in the Vacation Scholarship Readings, especially: Tom Baker's Presentation, the SKOS Primer, Megan’s AGIFT spreadsheet to RDF, and Jane Frazier's SKOS Manual. I found that this vocabulary work was easy to learn up to a certain degree, but very hard to master the more intricate workings of this process.

### Database

1. Download and Install Microsoft SQL Server Management Studio
2. When prompted to connect to server, input these fields (the password is ''password''):

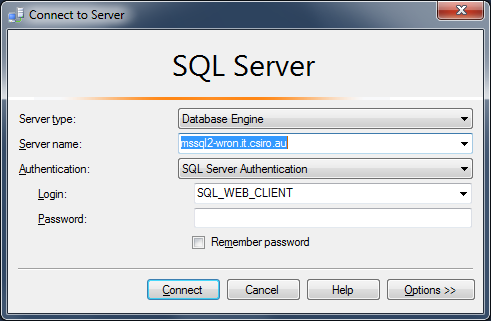


Figure : SQL Login

1. Find the Table folder in the Natsoil database:

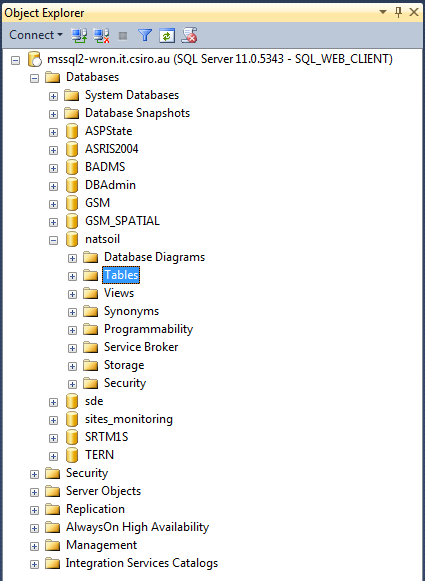


Figure : Menu Navigation

1. View the Top 1000 Rows of the table you're interested in (if there are more than 1000 entries, in the SQL code, just change ‘Select Top 1000’ to ‘Select’).

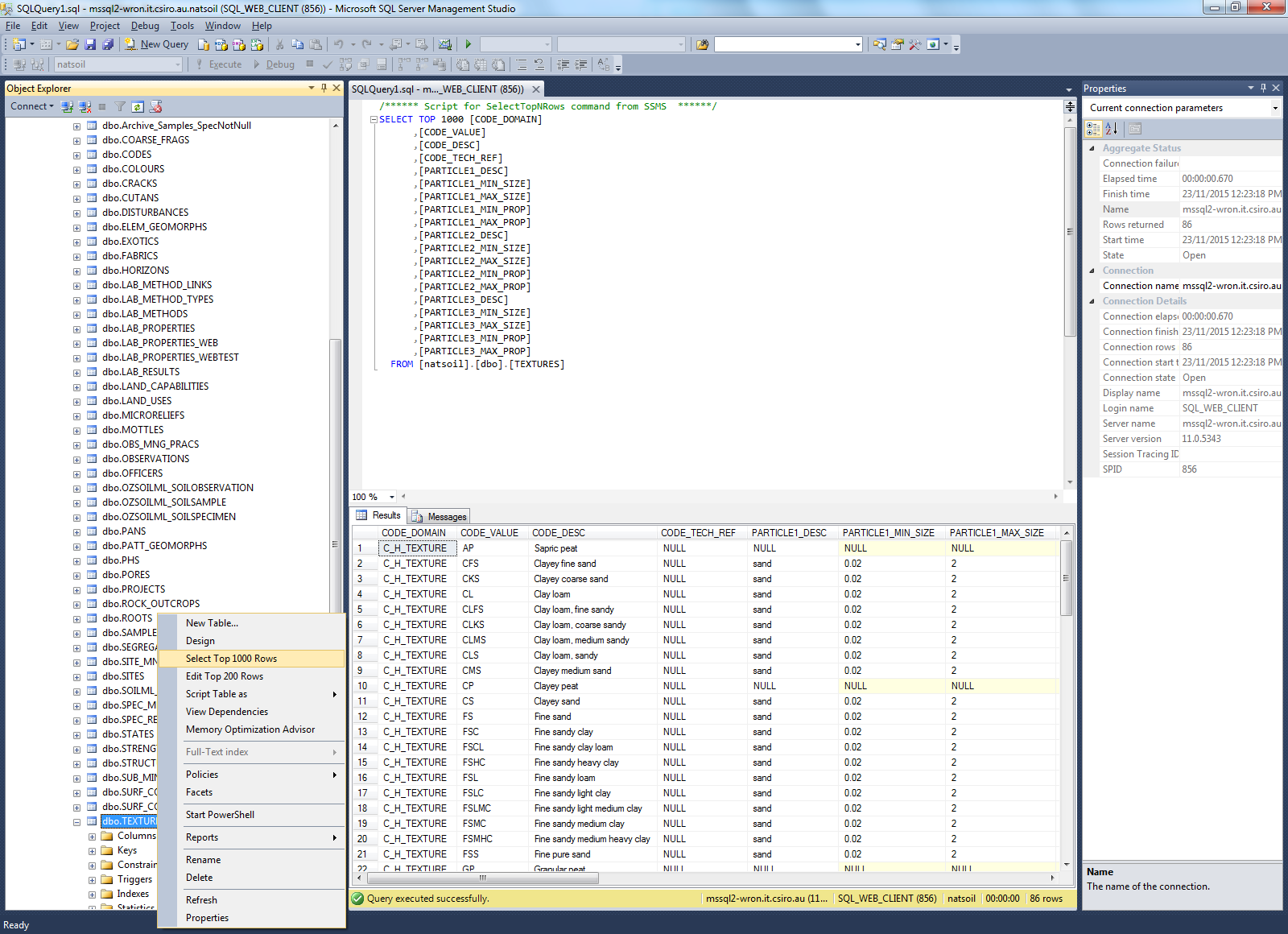


Figure : Top 1000 rows

1. Using the SQL code in the top half of the screen, eliminate the columns that are not needed. Then Save the Results as a .CSV:

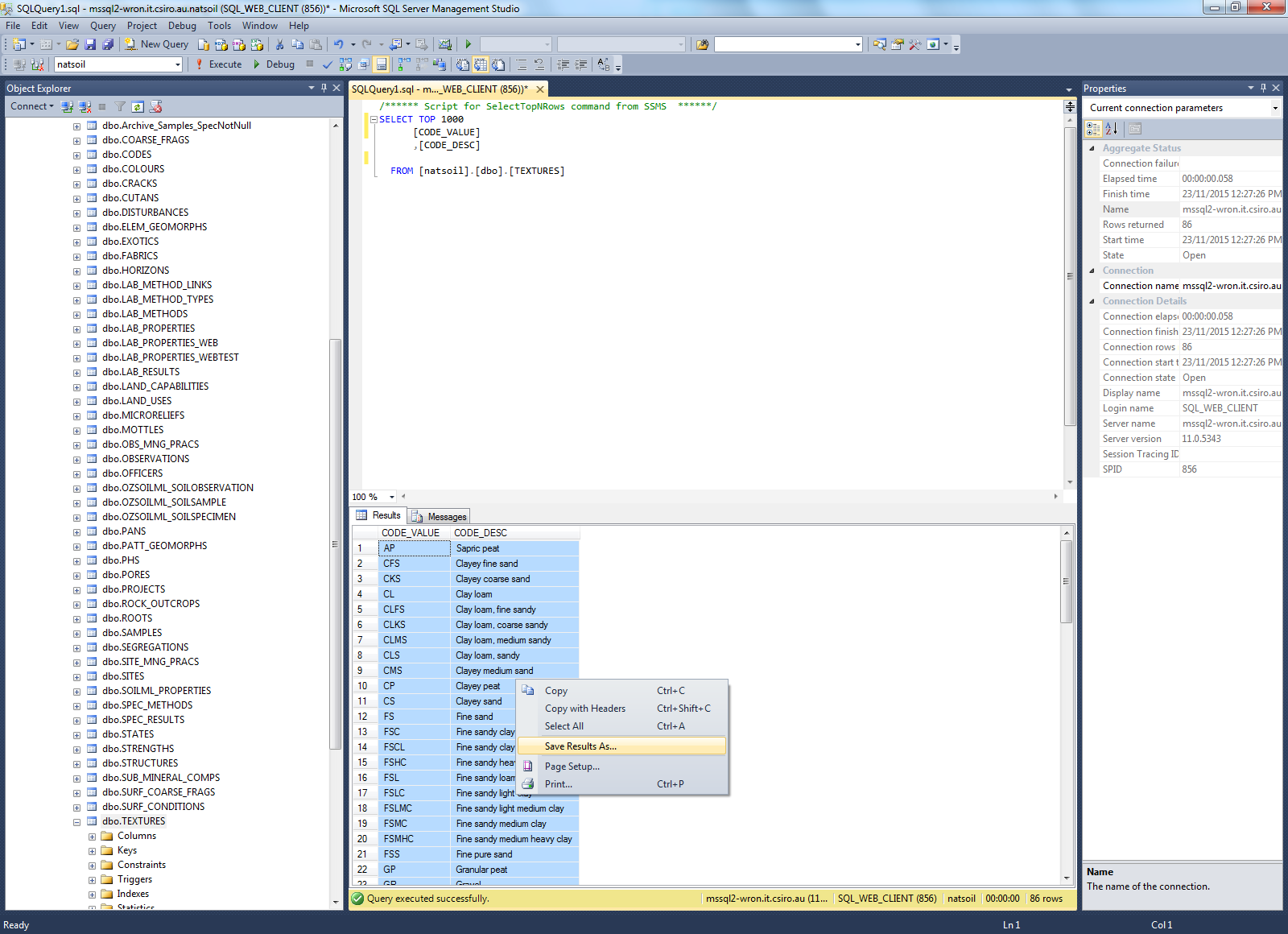


Figure : Save results

### Excel (CSV)

1. Establish best practice URI rules. In my case I used dashes for spaces, only used lowercase and used terms instead of codes
2. Decide on a base URI on which everything will be based off. Eg: http://registry.it.csiro.au/def/soils/texture/. Type it and Fill down.

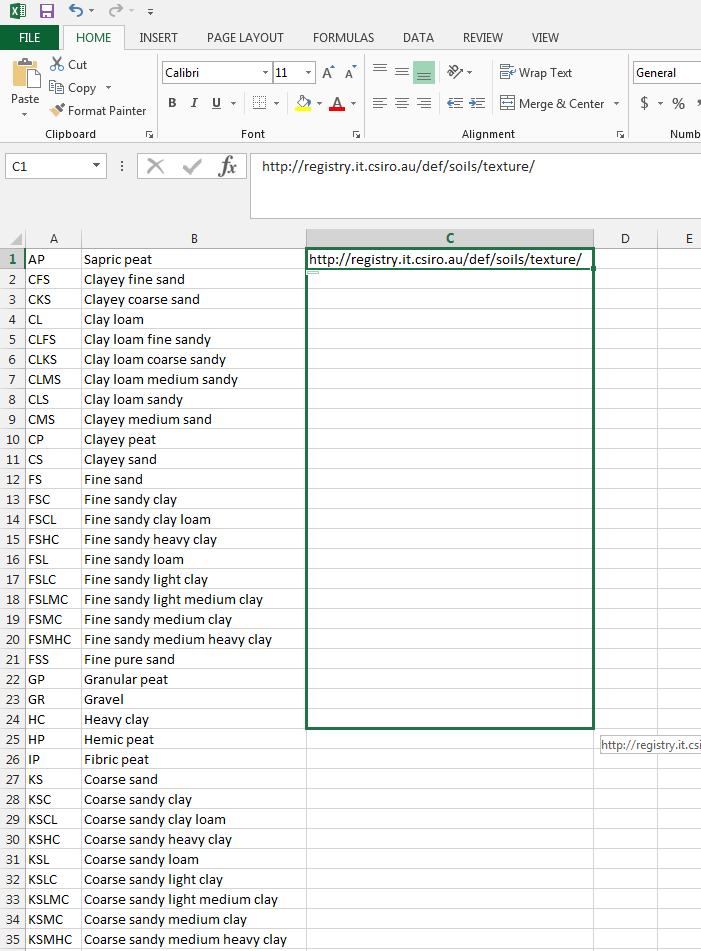


Figure : URI Filldown

1. Using a formula is the most efficient way to convert the terms in column B into a URI which follow the rules set out in point 1. The first function to use is the substitute formula, which replaces certain characters with something else. Eg: =SUBSTITUTE(B1, " ", "-") . Fill down.

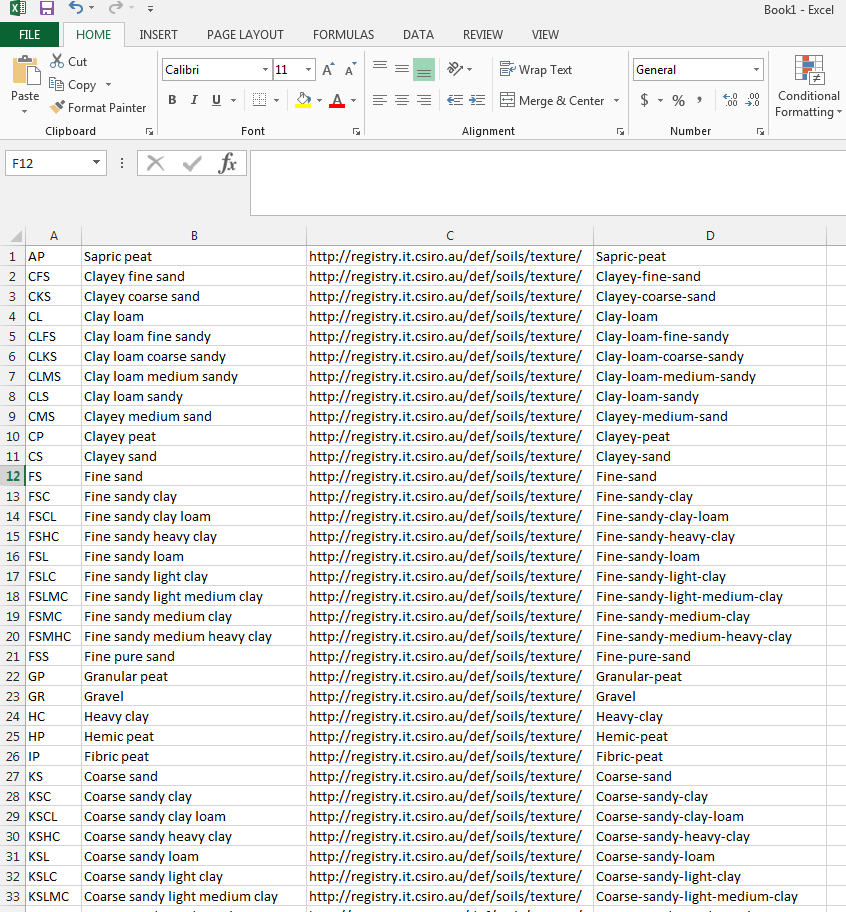


Figure : CSV Dashes

1. The next function will be the lower formula. This function changes everything in the cell to lower case. Eg: =LOWER(D1) . Fill down.

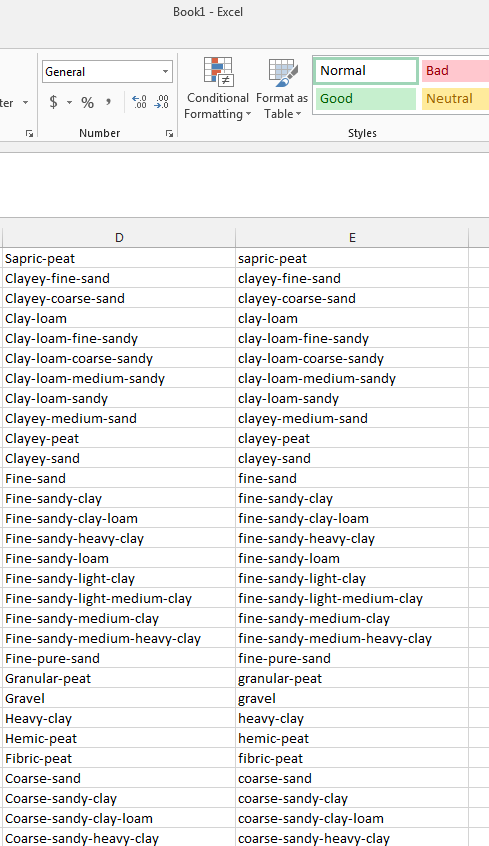


Figure : CSV Lower

1. To merge the base URI in column C with the unique URI prefix in column E, we need to use the concatenate function. This merges text from different cells and places it in a new one. Eg: =CONCATENATE(C1, E1) . Fill down.

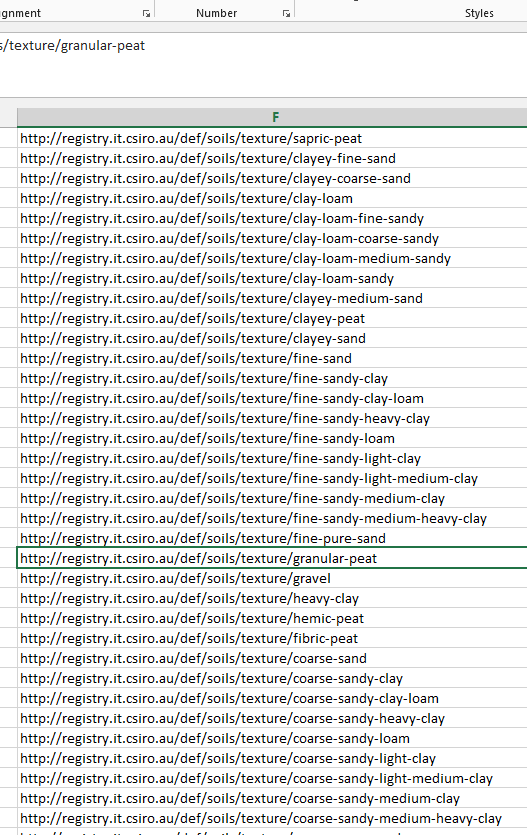


Figure : Concatenate

1. Next, we want to create the prefLabel. First, type 2 apostrophe's in a cell (Note: Not quotation marks). Fill down. Then type '@en' in the next cell. Fill down. We then want to concatenate these. Eg: =CONCATENATE(G1, B1, G1, H1) . Fill down.

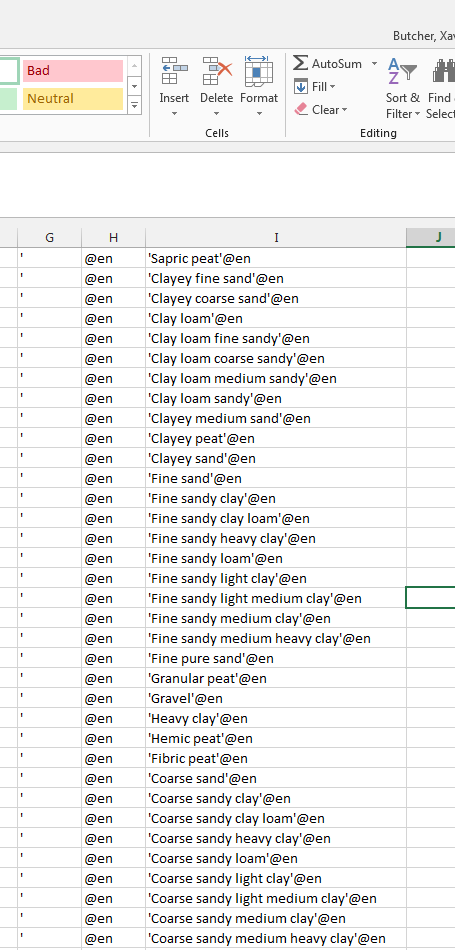


Figure : prefLabel

### RDF123

RDF123 is a powerful RDF editor. However, it is not user friendly at all. It may take some time to learn how to use the program properly.

1. In the prefix definition window, you will need to add or amend these definitions:

|  |  |
| --- | --- |
| **Title** | **URL** |
| Base | http://registry.it.csiro.au/def/soils/texture/ |
| mapBase | http://registry.it.csiro.au/def/soils/texture/ |
| Skos | http://www.w3.org/2004/02/skos/core# |
| Dct | http://purl.org/dc/terms/ |

1. The next step is to import the CSV file you created in excel. In the spreadsheet window, open your spreadsheet. Your screen should look like this so far:

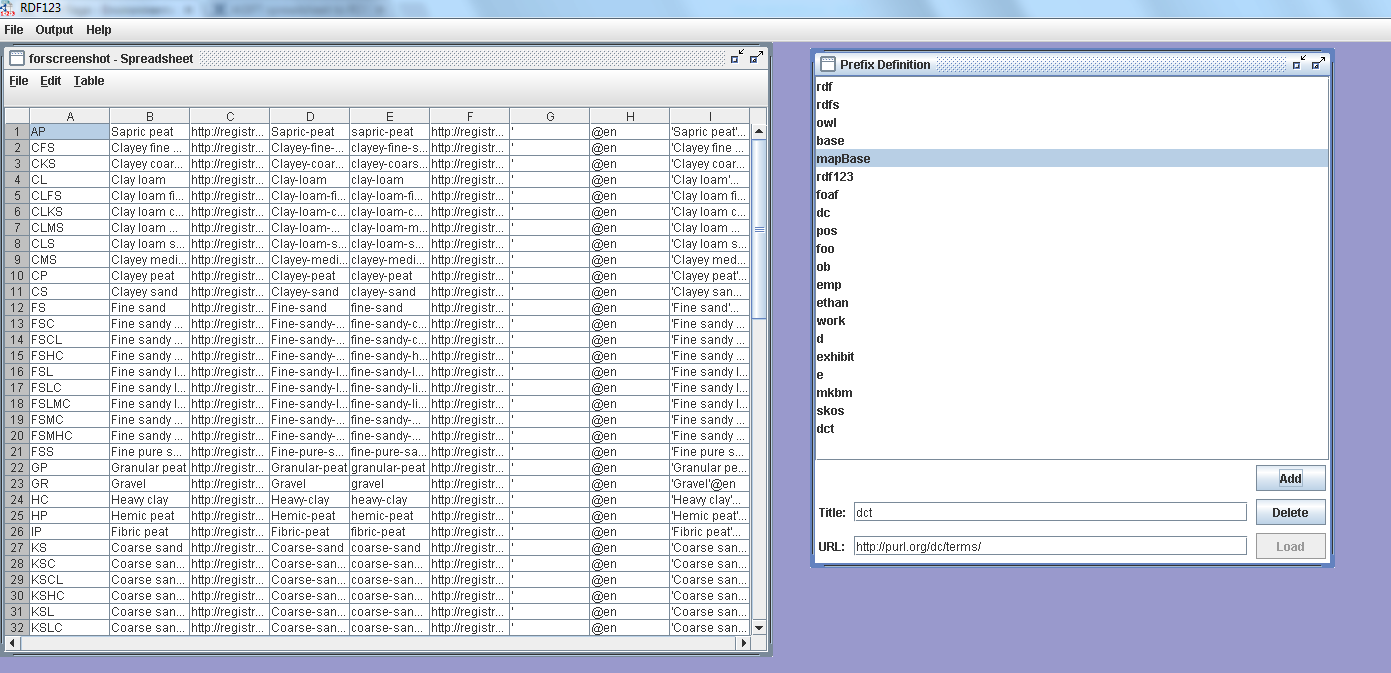


Figure : CSV Import

1. The next step is to use the 3rd window (map graph) to create a sort of mind map of what the data is trying to say. The first step is to create vertices and edges. If there is data already in the map area, just delete them, start with a blank canvas. In this case we will need 5 vertices and 4 edges. Don't worry about labels for now. The edges should radiate out of 1 central point to the other 4 vertices. It should look something like this:

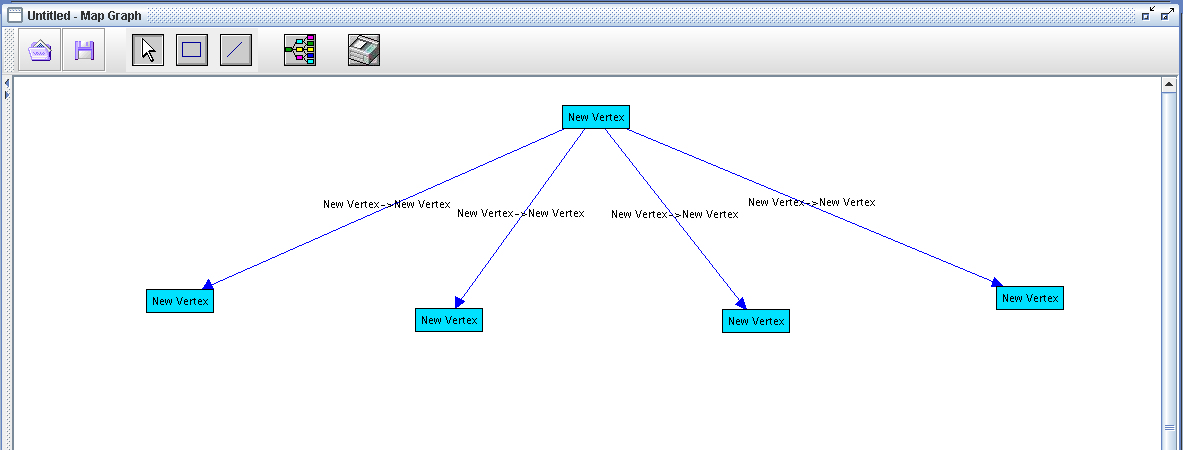


Figure : Blank Map

1. Now we need to start labelling the vertices. Firstly, the central vertex must be the unique URI. To tell the program that each individual URI is located in column F we need to use the notation "Ex:$6". The other columns you need are A, B and I (which are Ex:$1, 2 and 9 respectively). The remaining box must be called a "skos:Concept" to let the program know that each row is a separate concept.

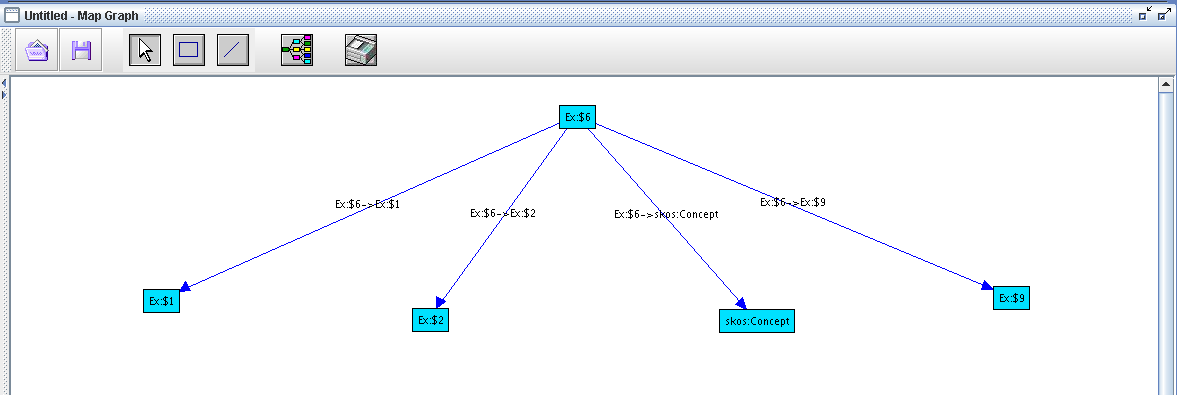


Figure : Map Labels

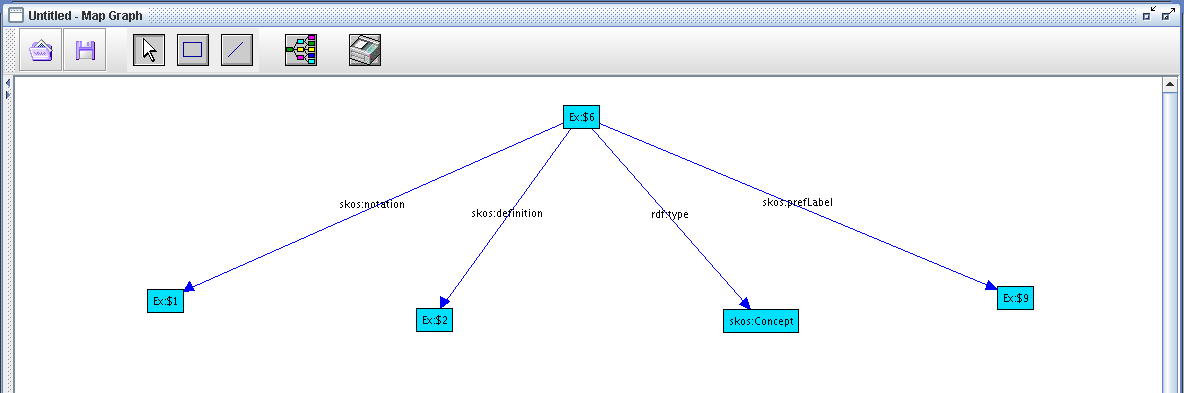
1. Next we must label the edges. Column 1 is a 'skos:notation', column 2 is a 'skos:definition', skos:Concept is an 'rdf:type' and column 9 is a 'skos:prefLabel'. Label the edges as such.
2. 

Figure : Map edges

1. You can now go to the overall window, and click the output menu at the top. Then click 'view spreadsheet in RDF'. Make the display type 'N3' as it’s easier to read. Copy this text output and paste it into notepad++.

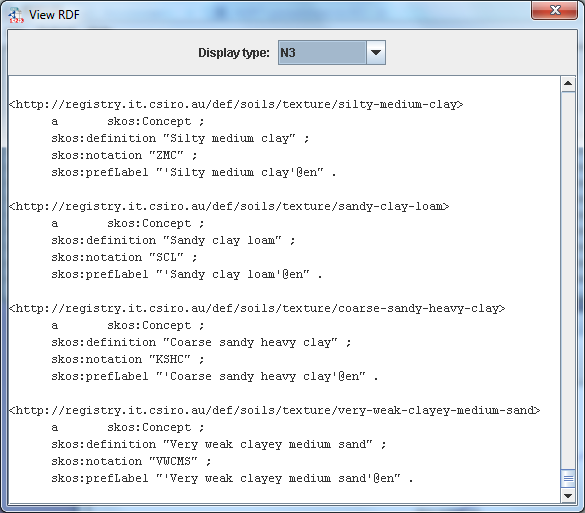


Figure : Output

### Text Editor (notepad++)

Unfortunately, the output from RDF123 is not usually error free. Some modifications to the text file will need to be made. As it is coding language, even the smallest discrepancy in the text will cause an error. Patience is key!

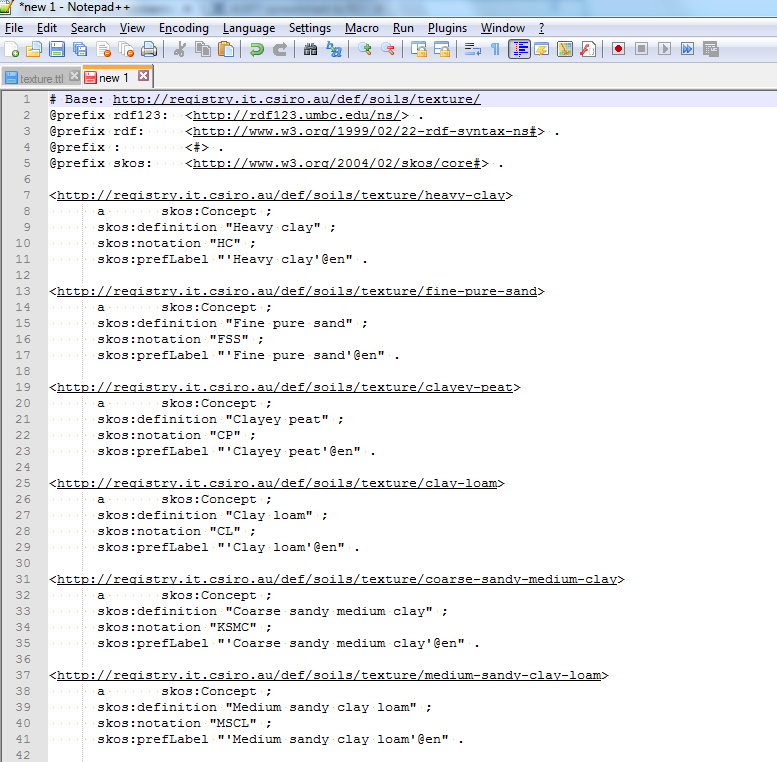
1. Initially it should look like this:  
   

Figure : N++ input

1. You're going to want to save it as a .TTL file so that it can be read by TopBraid.
2. The first thing you'll need to correct is in the first few lines:

|  |  |
| --- | --- |
| **Bad output from RDF123** | **Change to:** |
| # Base: <http://registry.it.csiro.au/def/soils/texture/> | #baseURI: <http://registry.it.csiro.au/def/soils/texture/> |
| @prefix : <#> . | DELETE |

1. The next thing that needs to be changed is that the prefLabel for each concept has not come across properly, due to RDF not recognising the '@en' as a separate suffix that isn't part of the concept name. To do this, use the Replace tool (in the Search menu). In this case for example, we will need to change <"'Heavy clay'@en"> to <"Heavy clay"@en>. This needs to be done throughout the whole document. The easiest way to do this is to replace all:

|  |  |
| --- | --- |
| **Find** | **Replace** |
| "'(quotation then apostrophe) | "(quotation) |
| '@en" | "@en |

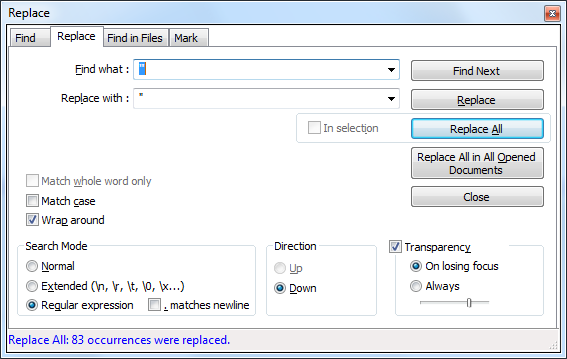


Figure : Replace text

1. This should be all that needs to be edited in notepad++, however there may be more depending on how it exported. It should look like this:

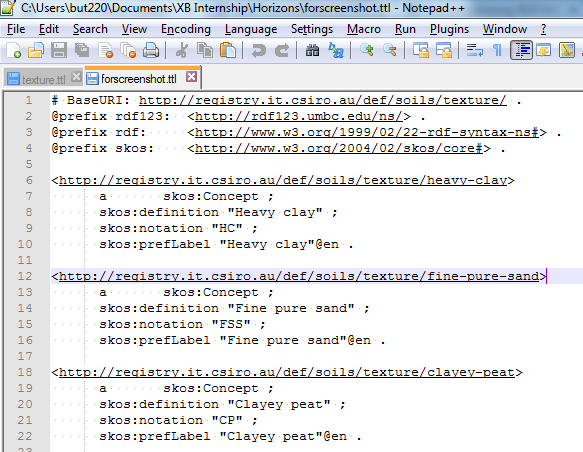


Figure : N++ Output

As a footnote, this table did not have a date created column. But if yours does, these are the steps needing to be completed, as dates come across poorly out of RDF:

**Editing**[**dct:created**](http://dctcreated/)**and**[**dct:modified**](http://dctmodified/)

Dates were not in the format required in RDF. There were a number of issues,   
1. Dates contained 0:00 at the end of the string  
2. RDF 123 picked up the text as a string  
3. Dates were in the form dd/mm/yyyy, but need to be yyyy-mm-dd

**Regex to fix these issue**

|  |  |
| --- | --- |
| **1.** | **Remove the 0:00 and recognize number as a date** |
| FIND | "([/0-9]+) 0:00" |
| REPLACE | "$1"^^[xsd:date](http://xsddate/) |

|  |  |
| --- | --- |
| **2.** | **Re-format so that date reads yyyy-mm-dd** |
| FIND | "([0-9]+)/([0-9]+)/([0-9]+)"\^\^[xsd:date](http://xsddate/) |
| REPLACE | "$3-$2-$1"^^[xsd:date](http://xsddate/) |

|  |  |
| --- | --- |
| **3.** | **Ensure that single digits have a 0 in front ( eg 9 --> 09)** |
| FIND | -([1-9])"\^\^xsd |
| REPLACE | -0$1"^^xsd |

### TopBraid Composer

1. To import your .TTL file into TopBraid, you must add the containing folder to the navigation panel in the bottom left. Then just double click on the .TTL file and it should open.
2. You must also import the SKOS core .TTL file and the Dublin core .TTL file (dc-1.1.TTL). Go to the imports tab at the bottom, and import the local file. From the properties namespace in the top right, drag the dct:description into the pane at the bottom.

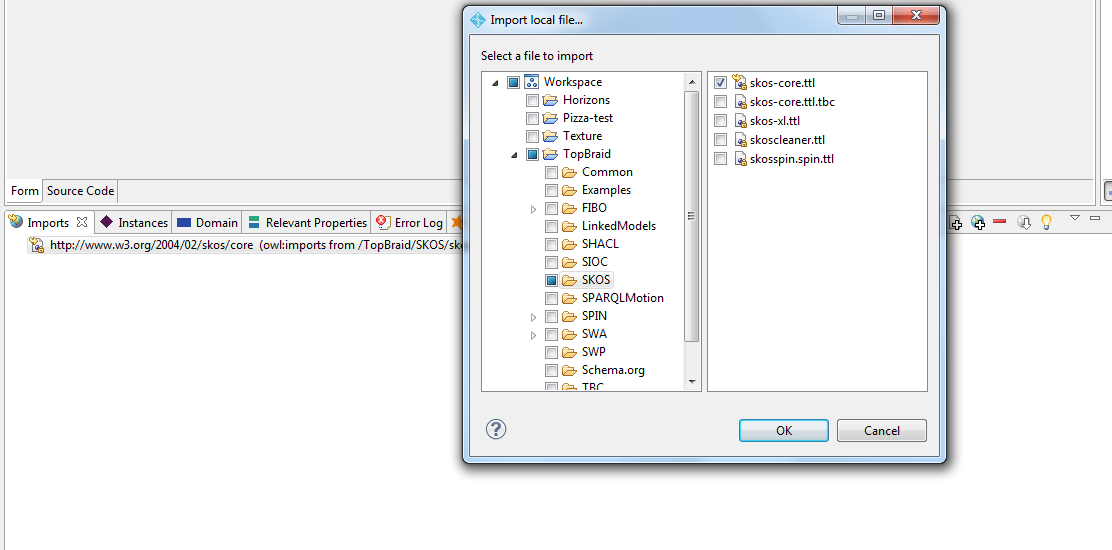


Figure : TopBraid Import

1. If the editing in notepad++ was done successfully, then each concept should appear in the Instances tab at the bottom, and in the skos:Concept tab in the Namespace at the top left.

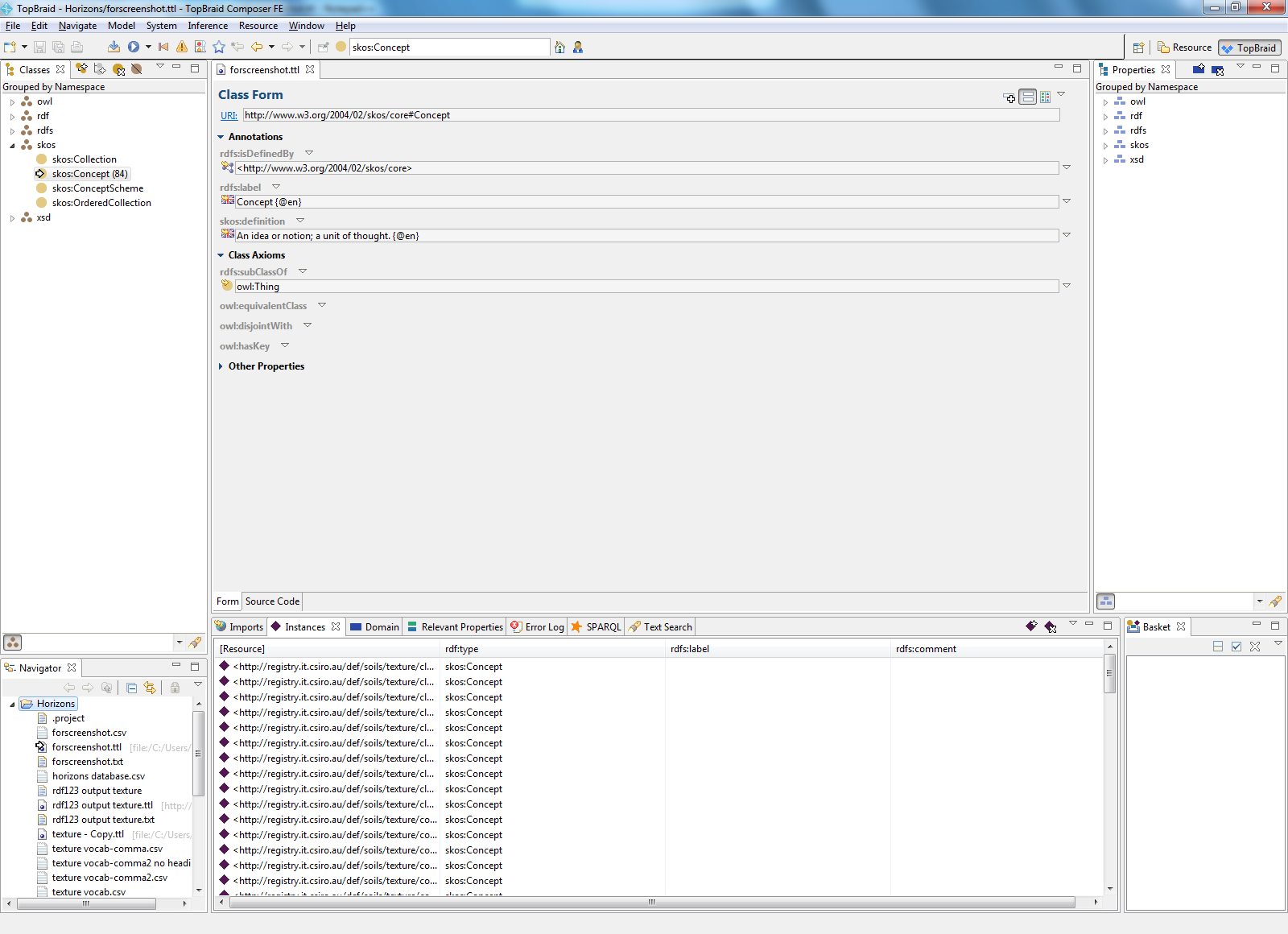


Figure : TopBraid Input

1. The next step is to add an rdfs:label to each concept. This is done by using SPARQL code found below:

INSERT{ ?s rdfs:label ?label }

WHERE {

{ ?s a skos:Concept . } UNION { ?s a skos:ConceptScheme . } UNION { ?s a skos:Collection . } UNION { ?s a owl:Ontology . }

NOT EXISTS { ?s rdfs:label ?l }

OPTIONAL { ?s skos:prefLabel ?pl . }

OPTIONAL { ?s dc:title ?t . }

OPTIONAL { ?s dct:title ?tt . }

BIND( REPLACE(str(?s), '^.\*(#|/)', "") AS ?localname)

BIND( REPLACE(str(?s), '[^/^#]+$', "") AS ?namespace)

BIND( REPLACE(str(?namespace), '(#|/)$', "") AS ?ns)

BIND( REPLACE(str(?ns), '^.\*(#|/)', "") AS ?nsfrag)

BIND ( STR ( COALESCE ( ?tt, ?t, ?pl, IF( STRLEN(?localname), ?localname, ?nsfrag) ) ) AS ?label )

}

1. The dct:description must also be added, using this SPARQL code.

INSERT { ?s dct:description ?desc }

WHERE {

{ ?s a skos:Concept . } UNION { ?s a skos:ConceptScheme . } UNION { ?s a skos:Collection . } UNION { ?s a owl:Ontology . }

NOT EXISTS { ?s dct:description ?d }

OPTIONAL { ?s skos:definition ?def }

OPTIONAL { ?s skos:scopeNote ?note }

OPTIONAL { ?s rdfs:comment ?com }

OPTIONAL { ?s skos:prefLabel ?plab }

?s rdfs:label ?lab .

BIND ( COALESCE ( ?def , ?note , ?com , ?plab , ?lab ) AS ?desc )

FILTER ( (datatype(?desc) = xsd:string) || ( lang(?desc) = "en" ) || (lang(?desc) = "EN") )

}

1. Once both of those search queries have been run, your instance panel should look like this:

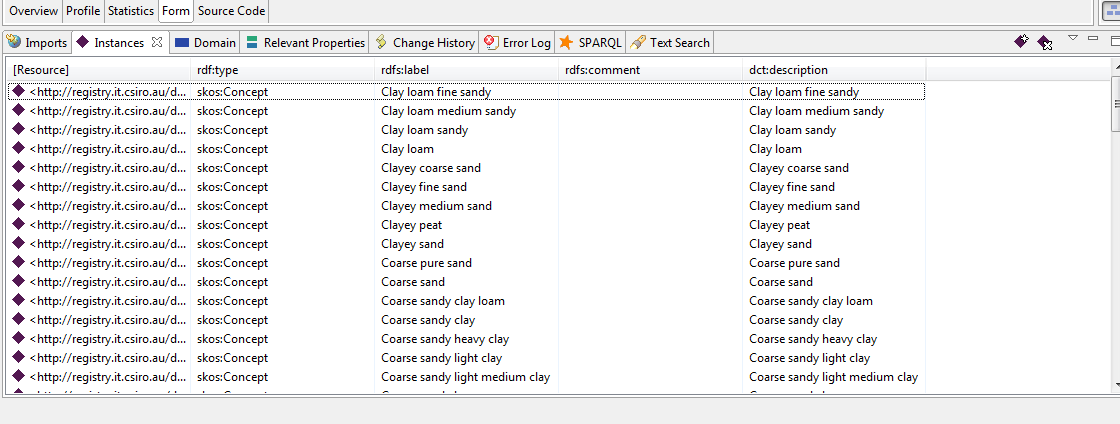


Figure : Description Added

1. The next step is to create a prefix for the concepts. Click the home button, and then click the overview tab. Add the prefix you would like to use, and then the base URI. I used texture in this case.

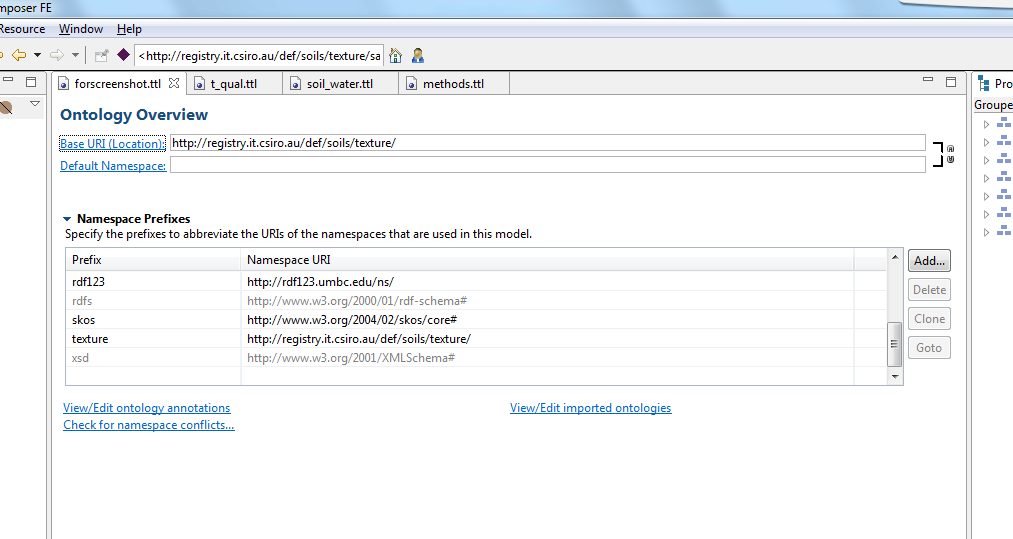


Figure : Prefix

1. Now is the time to do final quality control and checking of your concept data and make sure that all the properties you need have been filled. We must now do some final editing of the register in TopBraid before we can start uploading the vocabulary.
2. Click on the home screen, but this time navigate to the Form tab. Here we will need to add some necessary properties. The ones you'll need at a minimum are: dc:source, rdfs:comment, rdfs:label, dct:created, dct:creator, dct:description. Note: change the data type from string to date for date created (little white arrow at end of box). A purple diamond means the input should be a URI, whilst a blue square means the input should be a text string. If you do not have a URI for a particular field, you can get around this by firstly inserting a blank URI (<>) and then going to the source code and changing the <> to a " ". This will change the form to accept a text string instead. Your form should look like this after this has been completed:

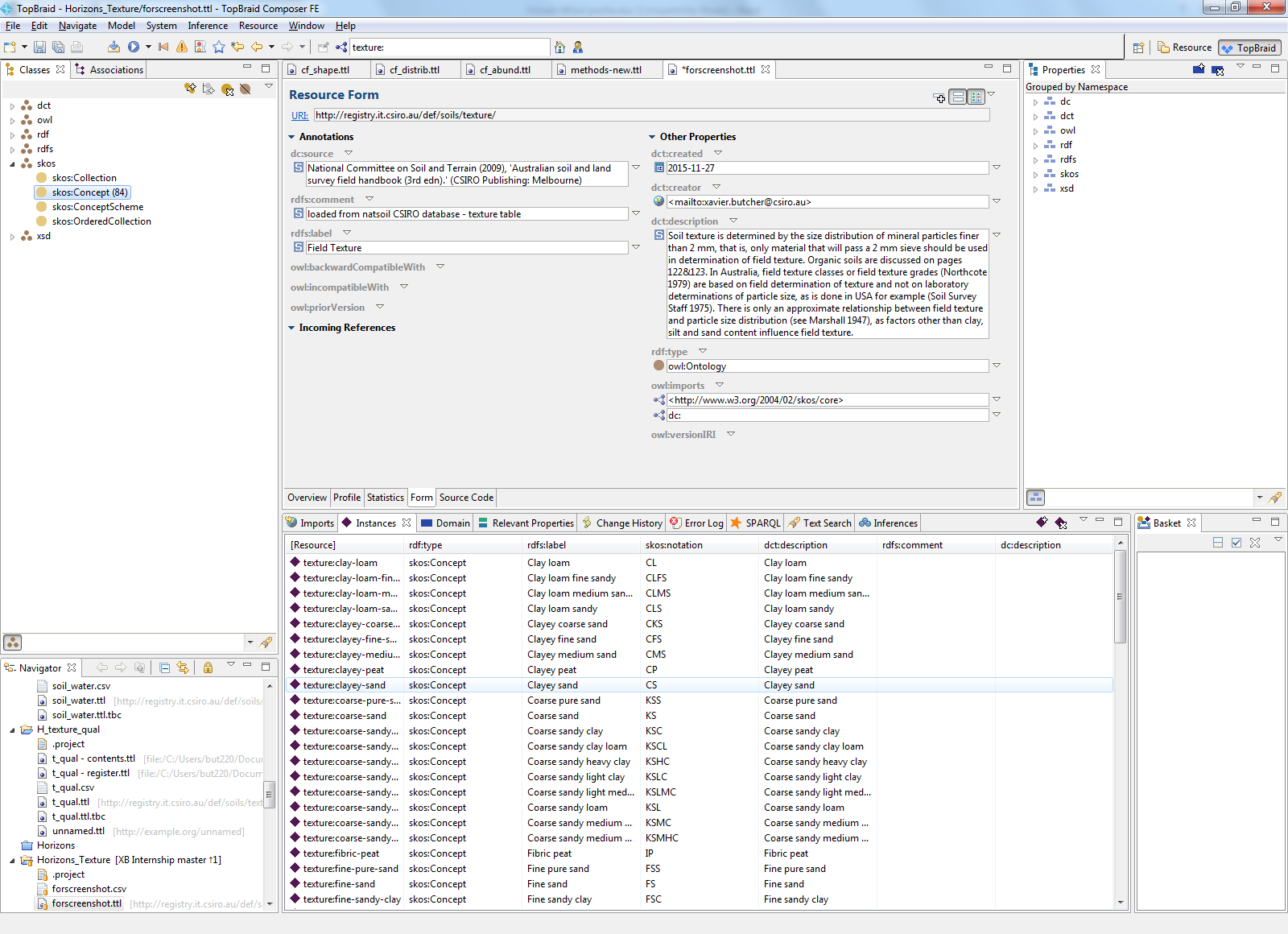


Figure : Registry Properties

1. This should be the end of the editing you need to do in TopBraid, however now we must use notepad++ again to do some final post processing.

### Post Processing (Text)

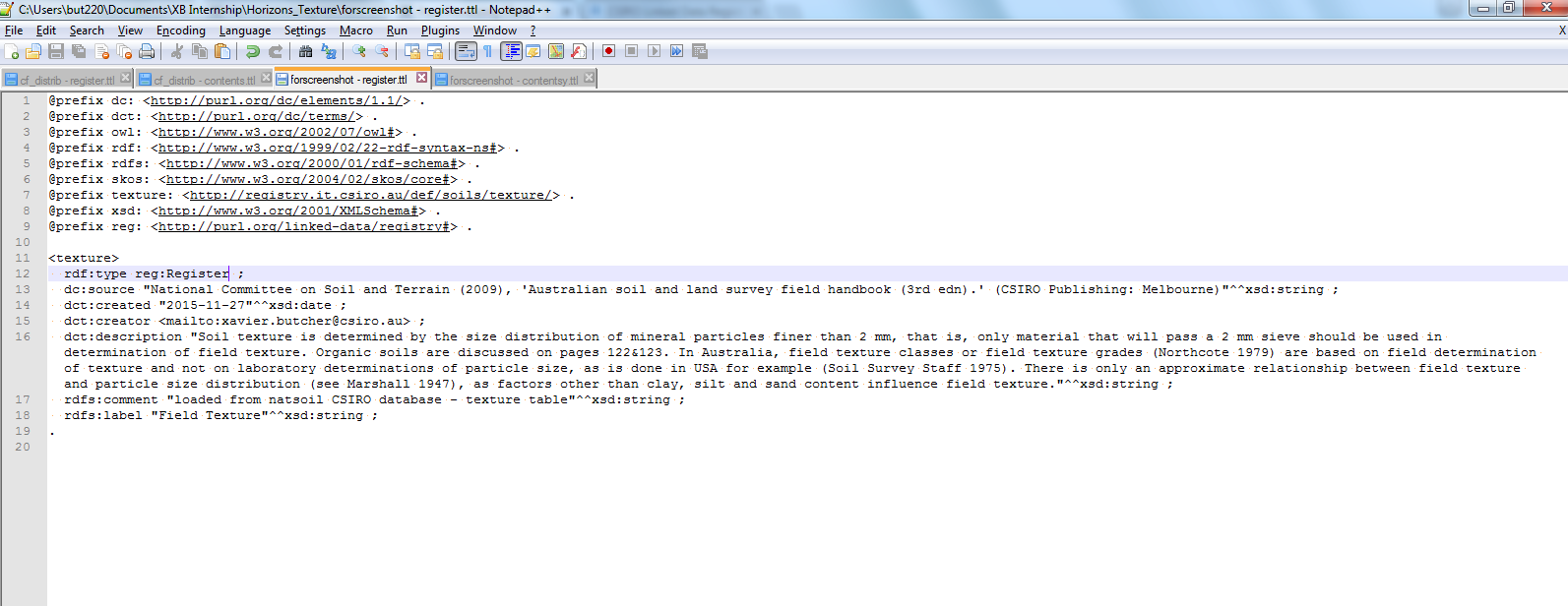
1. The first thing to do is create 2 copies of your .TTL file. One called registry, the other called contents.
2. We will start working with the register file. Firstly, delete the top 4 lines (baseURI, imports 1 and 2 and a blank line).
3. Next, we will delete all of the 'contents', leaving only the register information.
4. Following this, delete line 5 (the rdf123) prefix. Then add another prefix: @prefix reg: <[http://purl.org/linked-data/registry#](http://purl.org/linked-data/registry)> .This just tells the program that this particular file is part of the registry.
5. We will then change "owl:ontology" to "reg:register".
6. We will then change line 11 from "texture:" to "<texture>. This is just the end of your base URI.
7. Then we will delete lines 19 and 20, the 2 owl imports.
8. That is it for the registry file, we will now move onto the contents file. The final registry file should look like this:  
   

Figure : Registry

1. Open the contents file.
2. Much like the registry file, we will delete the top 4 lines. We will also delete the prefix rdf123 line. But this time we will delete the registry information, and keep the contents information.
3. We then need to do a bulk search and replace. We need to change "texture:clay-loam" to "<clay-loam>". The regular expression code you need for this is:

|  |  |
| --- | --- |
| **Find** | **Replace** |
| texture:([a-zA-Z0-9\_\-]+) | <$1> |

1. The text editing should now be done. The contents document should look like this:

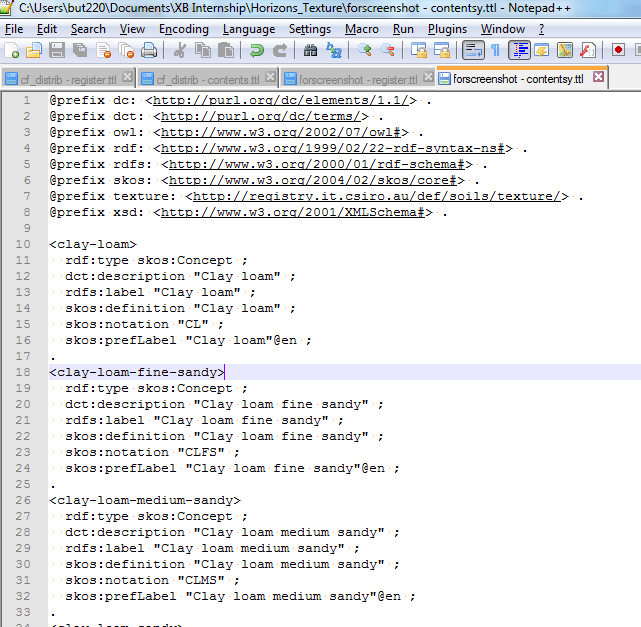


Figure : Contents File

### Publishing

1. Confirm your registry and contents files are correct, as once uploaded to the Linked Data Registry (LDR), it is hard/impossible to remove/add/modify.
2. Navigate to your registry location. You will need to create an account/login if you haven't already. Mine is: http://registry.it.csiro.au/sandbox/student/xavier
3. The page should look like:

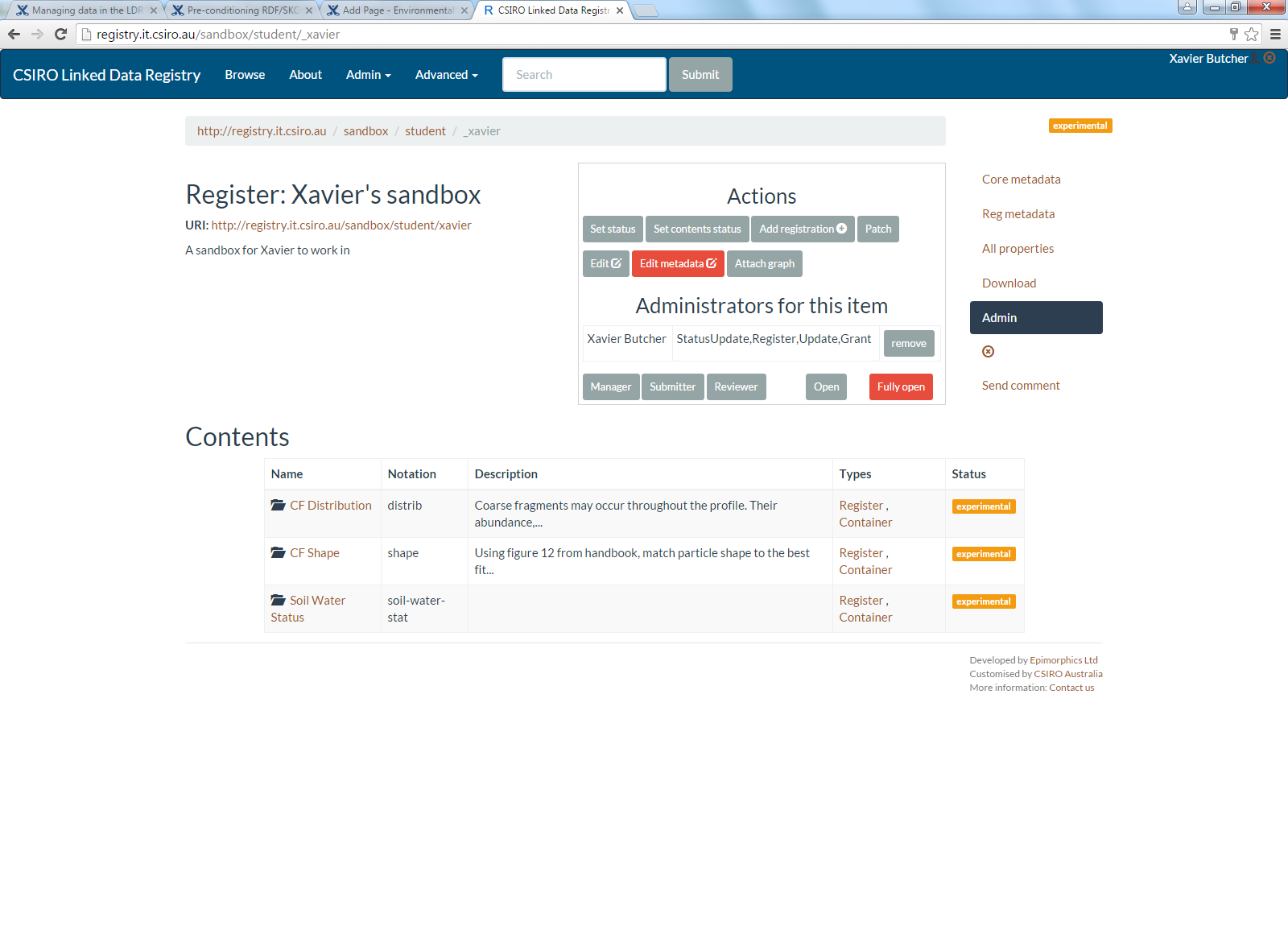


Figure : Navigation

1. Then click on admin, and select add registration.
2. Next, click upload, and choose your registry file.
3. It should look like this:

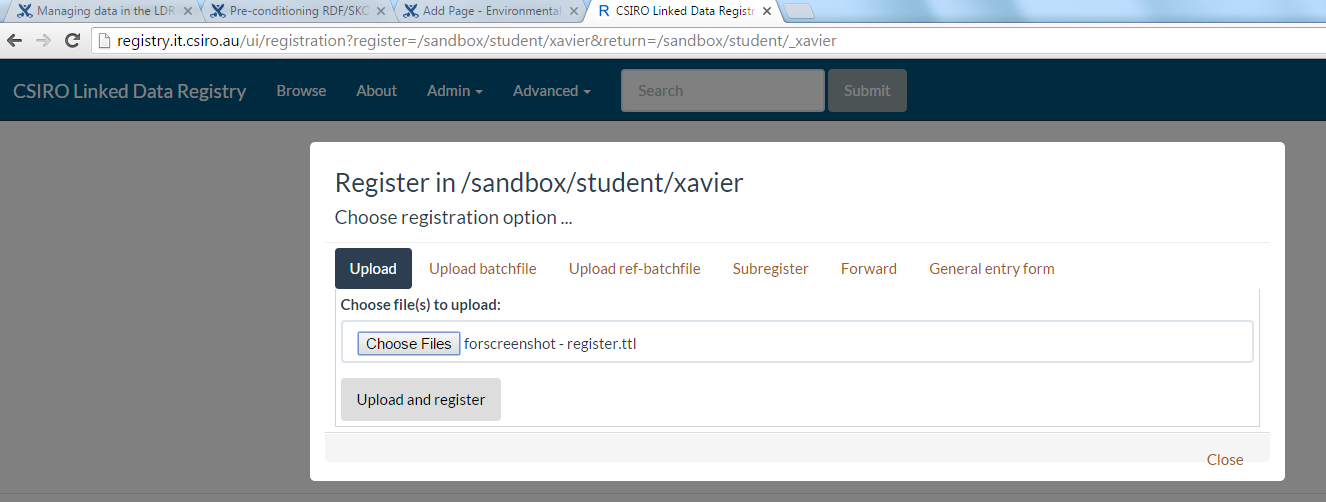


Figure : Register upload

1. Once uploaded, you should see the new register appear. Click on it, and you should see the register you uploaded, with labels and descriptions. However, there shouldn't be any contents - yet.
2. When inside the register, using the same method as before, upload the contents file. It should look like this:

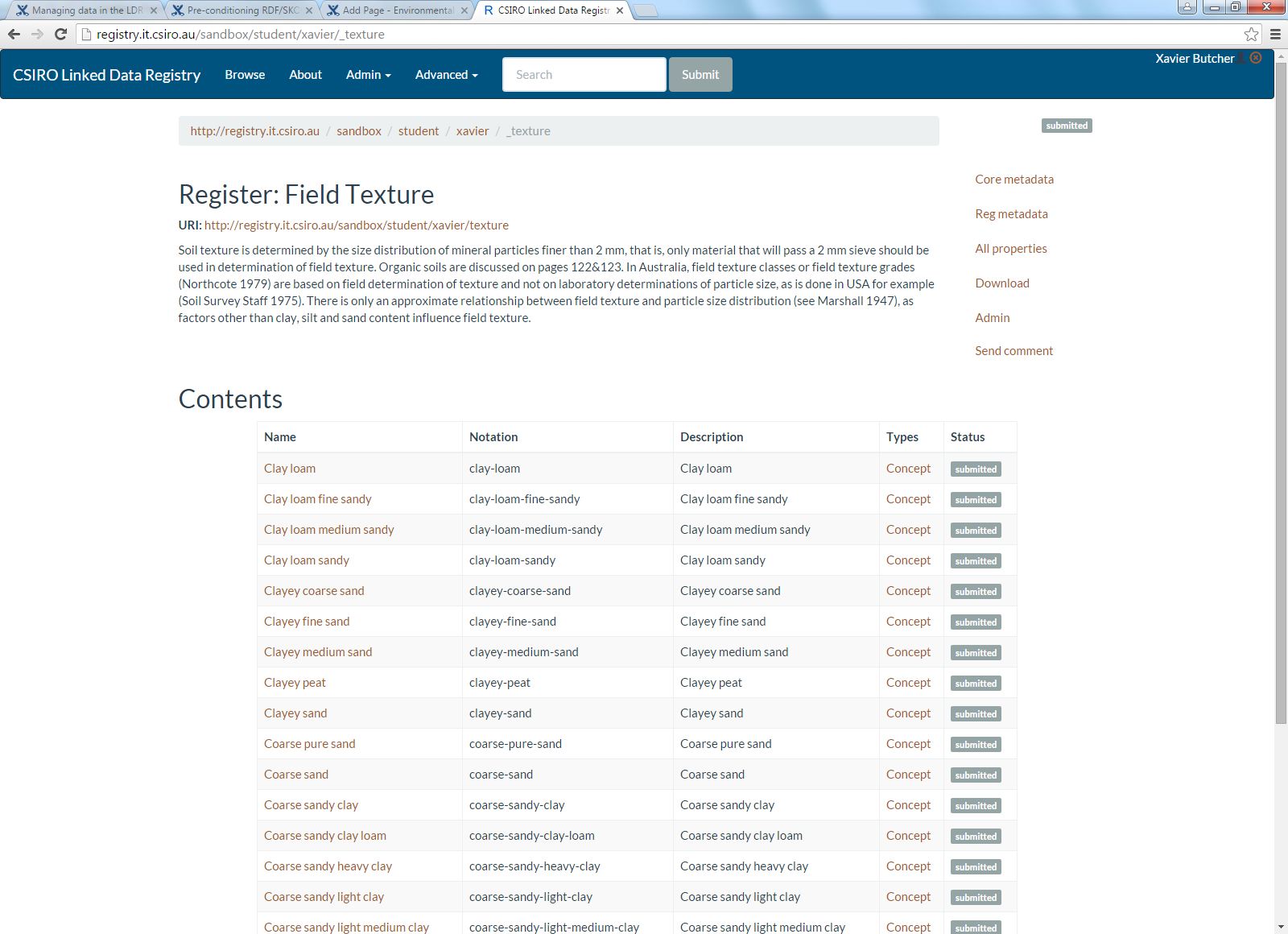


Figure : Contents Upload

1. You will notice the register and all the contents say submitted. This means only you can see them. To 'turn on' your vocabulary, you must make the status of both the register and the contents "Experimental".
2. This is done by clicking on admin, then 'set status'. A box should come up; click on the yellow experimental button. Do this process again for the 'set contents status' button. It should look like this:

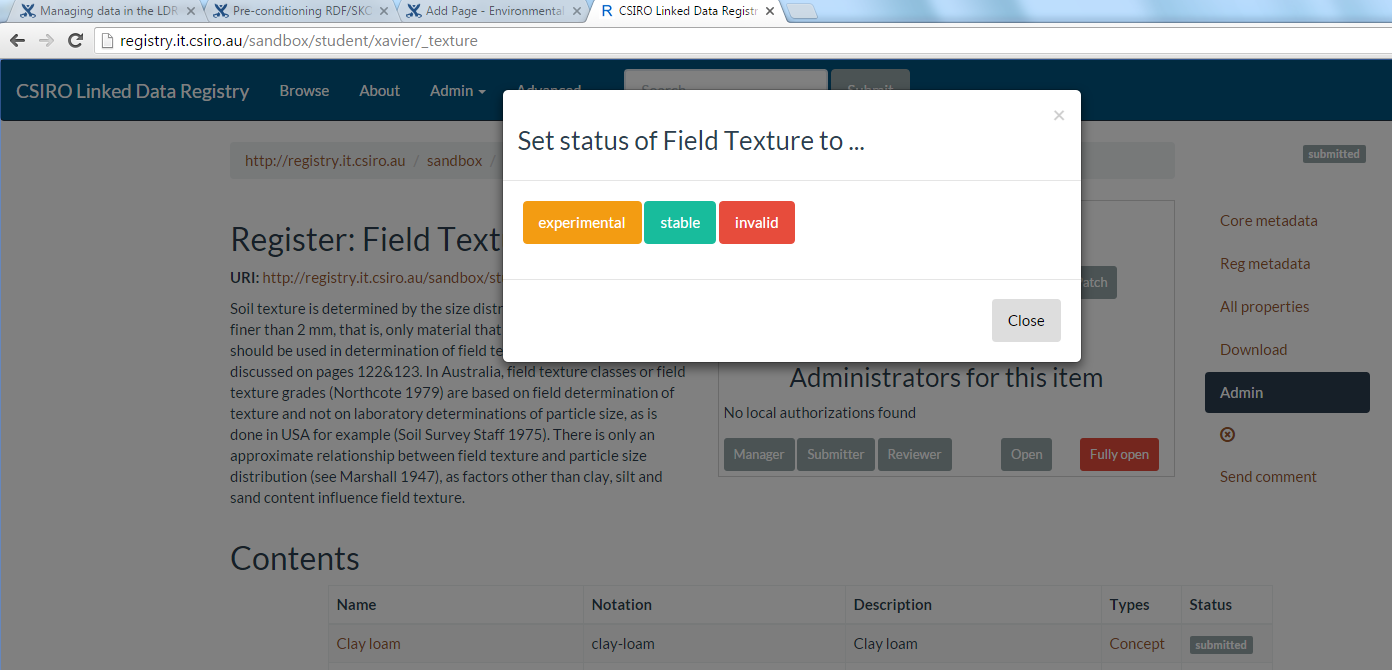


Figure : Set Status

1. Your vocabulary is now complete!

# Results

I have completed 27 vocabularies and counting. These are:

|  |  |  |
| --- | --- | --- |
| CF Abundance | Soil Taxonomy | Lab Methods |
| Plasticity Degree | Carbonate Effervescence | Moisture Status |
| Stickiness | CF Distribution | CF Size |
| ASC | CF Shape | Crack Width |
| ASC Confidence | Soil Water Status | Pore Abundance |
| ASC Family | Texture Qualifier | Root Abundance |
| Drainage | Texture | Root Size |
| Technical Reference | Water Repellence | Strg Class |
| Plasticity Type | Lab Units | OP Methods |

These can all be seen in full on the CSIRO Linked Data Registry.

# Discussion

## Key Issues

When exporting from CSV to RDF123, some CSV files inexplicably exported as Tab Separated Values instead of Comma Separated Values. This is means RDF123 won’t be able to read it. All that is needed to fix it is opening the CSV in notepad, and using the find and replace function to replace tabs with commas.

When trying to get TopBraid to prefix a URI, often times it would not prefix correctly because of an extra slash at the end of the URI. I don’t know how this error comes about, but it’s easy enough to fix – just delete the slash.

When adding the @en tag to a prefLabel, it is supposed to sit outside of the quotations of the label itself. However RDF123 recognises the whole thing as a label and puts it in quotations. There is no permanent fix I can see for this as that is how the software was designed. The temporary fix is just using the find and replace function in notepad to remove them.

Some vocabularies had information regarding the range of the field. For instance Crack Width. The issue was how to store that information in a vocabulary. I chose to use the xsd:minInclusive and xsd:maxExclusive properties, but it is possible there is a better solution.

The issue of copyright must also come into consideration. Many of the definitions of the concepts were taken verbatim from the soil handbook, so permission must first be granted by the publisher before this can proceed.

In the LDR, for some reason the system would take the arbitrary notation I had used and make that part of the URI instead of the proper name I had given it. Eg: /xavier/abund instead of /xavier/cf-abundance.

There were times when a missing definition needed to be added for a particular method or element or concept. Having done no work on soils, and only a mediocre chemistry and physics background, this was challenging – even if complex textbooks were available.

There were some instances where 2 tables had identical or very similar data. I chose to merge these tables where feasible as this not only creates less work for me, but is much more concise and causes less confusion.

There is also an issue with URI location. My vocabularies were initially stored in a sandbox registry – in case the intern stuffed something up. There was some confusion about whether the URI of an object should be pointing to the current sandbox location, or to the anticipated true location. The latter was chosen.

## Specific Issues – Lab Methods

Lab methods is the most difficult vocabulary I had to create. It has around 600 data entries across several columns. However, the number of fields that classified the data was not sufficient. So I had to add in several more to get a functioning vocabulary. These included Full Name and Object of Interest.

The biggest issue with Lab Methods was the fact that the only unique identifier was a random arbitrary code. This is not user friendly at all. So for each of these 600 entries, I created a unique identifier. This was done by taking the object of interest of each method, and assigning it a number. Eg: Aluminium-1. I chose this method because I figured that a common use of the vocabulary will be for if a user wants to know all of the different methods of measuring aluminium.

Because that is the aim I’m looking to achieve, the best way to classify these is not only by URI, but to organise these into collections of object of interest. These collections make it easy to search and navigate to this information.

I decided to use the arbitrary lab code as the prefLabel, which I feel is only a temporary solution. I do not yet know how the domain manager would like to classify it, but I would recommend some new label that caters to people who want to search by name as well as those who want to search by lab code.

There was also the issue of units of measure. There were 2 measures that did not have any vocabulary reference anywhere online. So I had to create these and add them to the environment database.

Another property I added was op:method. This is so users can search by method type rather than object of interest. Eg: Volumetric Analysis.

As requested, I created a diagram showing which fields from the database mapped to which fields in the final vocabulary. This is important as a check to make sure all information is transferred.

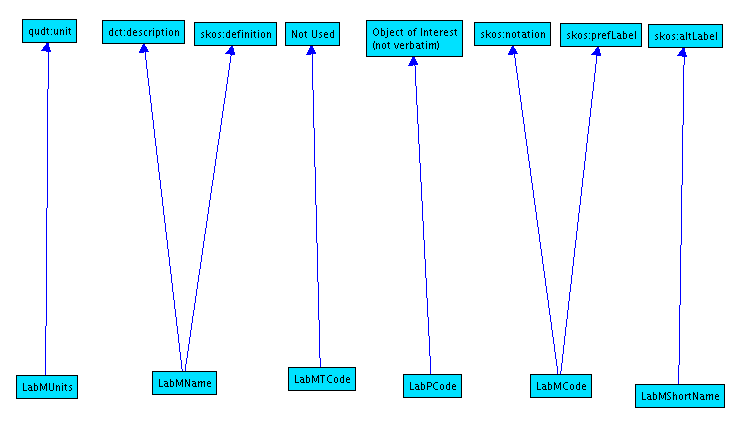


Figure : Lab Method Mapping

## Future Developments

I thought it would be a good idea to outline where I think future developments and ongoing work could exist based on the project I’ve already done.

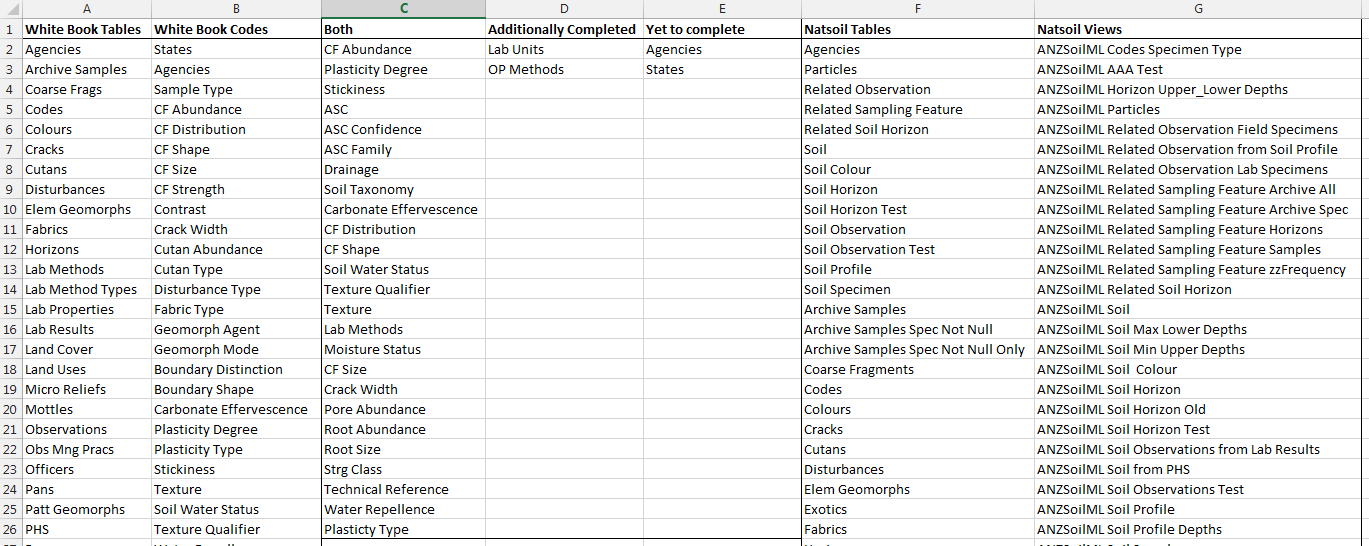
The first thing that needs to happen is to get the online database up to scratch. There are 185 tables in the reference book, but only around 30 of these are in the database. If my project is to be continued on, this needs to happen. I created a spreadsheet to display this information – in a Venn-diagram type arrangement. 

Figure : Venn diagram of book vs database

Secondly, the process of creating a vocabulary from a database, whilst difficult to learn initially, is rather repetitive in nature. I would recommend that a computer coder (perhaps in the form of another intern) be brought in to create a script to automate this process. I feel the process should almost certainly be able to be automated, and if it is possible, will allow for efficient conversion from database to vocabulary. The script could also be used for other domains outside of soil, as the process is generic.

Finally, the worst piece of software to use in this process is RDF123. It is not very user friendly, and there are many issues with it that hinder what I am trying to do such as the @en tag I mentioned before – and not to mention the spontaneous freezing of the application. I would recommend that another project could be to create a newer, more stable, and more user friendly and efficient program to create RDF files from a CSV.

# Conclusion and Personal Reflection

Coming from a Surveying background, and having little-to-no knowledge of vocabularies and informatics, this project has been very helpful in developing my skills on informatics technology and helping me understand what is it the Environmental Informatics team do. I have learnt about vocabularies, SKOS, RDF and even a bit of SPARQL. I also learnt how to use a few new programs like RDF123 and TopBraid. This project has definitely broadened my horizons. I have seen that some of the skills learnt on my time here will be very relevant to my future endeavours. I hope that my work is useful to CSIRO in some way, and that it either gets applied or contributes indirectly to future projects. I am proud of the quality and quantity of work I have achieved during my internship.

If I was to criticize this project, it would be to note the routine nature of the methodology used to attain the vocabularies. It is indeed repetitive and time consuming. I would also note that the period of time where I was left with limited or no supervisors, especially early on in the piece where I needed the most guidance and technical questions was sub-optimal. This was due to prior commitments which took the supervisors off site for much of December.

# Acknowledgements

I would especially like to thank Jonathan Yu, Simon Cox and Bruce Simons for their continuous input and guidance on this project. Their help was much appreciated, and I’m sure that they are very much looking forward to not having to answer my rudimentary questions or verifying my SPARQL code. I would also like to thank Luis Neumann for helping with all the initial induction documents and general administration throughout the internship. Lastly of course, I need to thank Section Leaders, Branch Heads and Recruitment for allowing me to undertake this internship.