

BINF 8211/6211
Design and Implementation of
Bioinformatics Databases
Lecture #18

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

Review Relational Model

Review

- MySQL command line functions:
 - for importing data?
 - for exporting data?
- Data ETL
 - What does ETL stand for?
 - Briefly describe each of the components.
- In “T” what are the four components to look for?
 - What is an example of a data inconsistency
- What is RDF and why was it created?
 - Why does it not work well for science.

- 1) mysqlimport --columns='head -n 1 \$yourfile' --ignore-lines=1 dbname \$yourfile
- 2) SELECT * FROM Cars INTO OUTFILE '/tmp/cars.csv' FIELDS TERMINATED BY ',';
- 3) Extract, Transform, Load

Acquisition, QC, Import

- 4) Transform can be thought of as four components: Your QC should look for incomplete data, incorrect data, incomprehensible data and inconsistencies in the data.

- 5) Data where the units do not align, variation in coded meaning.

Illumina read FASTQ scores are different depending on the software version that was run. Same code can imply a different code.

- 6) Resource Description Framework was developed by the W3C to have a manner in which to build a Semantic Web.

- 7) Known issues include these issues include ‘too few entailments’, ‘contradiction classes’ and ‘size of the universe’

Lecture Schedule

- MySQL tools
- OWL
- OBO part II

SQL Topics (MySQL)

- String tools
- Date and Time
- Math functions

MySQL String tools

- Compare strings `strcmp(string1,string2)`
- Convert to lower case `lower(string)`
- Convert to upper case `upper(string)`
- Left-trim whitespace (similar right) `ltrim(string)`
- Substring of string `substring(string,index1,index2)`
- Encrypt password `password(string)`
- Encode string `encode(string,key)`
- Decode string `decode(string,key)`

MySQL (Data Time)

- Get date
 - curdate()
- Get time
 - curtime()
- Extract day name from date string
 - dayname(*string*)
- Extract day number from date string
 - dayofweek(*string*)
- Extract month from date string
 - monthname(*string*)

MySQL (Math Functions)

- Count rows per group
`COUNT(column | *)`
- Average value of group
`AVG(column)`
- Minimum value of group
`MIN(column)`
- Maximum value of group
`MAX(column)`
- Sum values in a group
`SUM(column)`
- Absolute value
`abs(number)`
- Rounding numbers
`round(number)`
- Largest integer not greater
`floor(number)`
- Smallest integer not smaller
`ceiling(number)`
- Square root
`sqrt(number)`
- n th power
`pow(base,exponent)`
- random number n , $0 < n < 1$
`rand()`
- sin (similar cos, etc.)
`sin(number)`

The Open Biological /Biomedical Ontologies or OBO Foundry is a collaborative community.

- The OBO Foundry
 - <http://www.obofoundry.org/>
- The NCBO BioPortal is a repository of information about biomedical ontologies (and related tools).
<http://bioportal.bioontology.org/>
 - Concepts are given unique identifiers, assigned by UMLS*
 - UMLS* is the Unified Medical Language System – the National Library of Medicine produces and distributes information about it.
 - For systems that create, process, retrieve, and integrate biomedical/health data and information
 - MetamorphoSys application: a Java tool for local application of the 3 components of the UMLS Knowledge Sources
 - Documentation: <http://www.ncbi.nlm.nih.gov/books/NBK9675/>

3/31/16

8

The tools help you build well-formed and correct structures – they help you find mistakes, and also layout existing structures, and compare existing structures.

is a collective of ontology developers that are committed to collaboration and adherence to shared principles. The mission of the OBO Foundry is to develop a family of interoperable ontologies that are both logically well-formed and scientifically accurate. To achieve this, OBO Foundry participants voluntarily adhere to and contribute to the development of an evolving set of principles including open use, collaborative development, non-overlapping and strictly-scoped content, and common syntax and relations, based on ontology models that work well, such as the Gene Ontology (GO).

Some of the NCBO ontologies are private – a subscription is required. It uses a REST API.

Patient records, scientific literature (like MeSH), guidelines, public health data.

Associated resources include the Semantic Network, the SPECIALIST lexicon, and Lexical Tools.

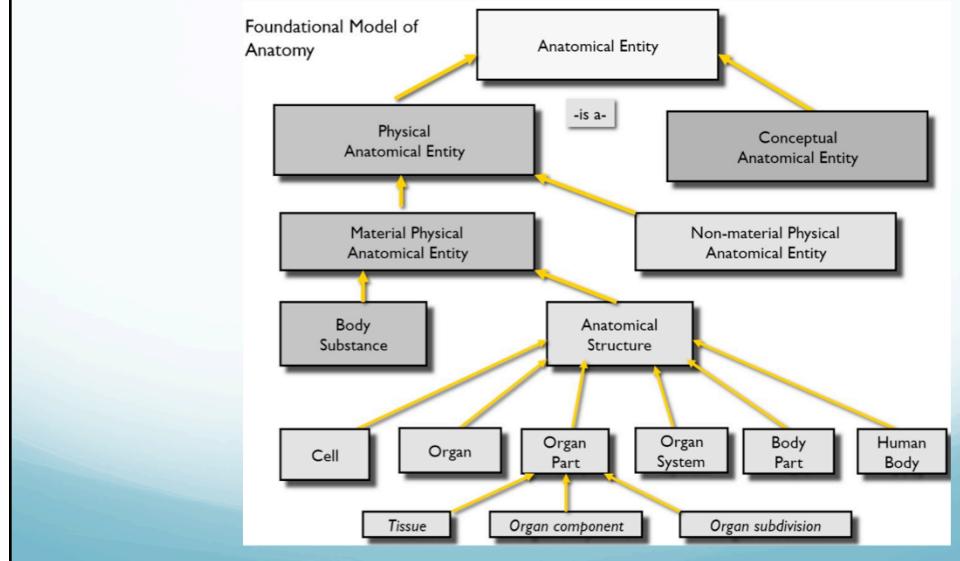
There is a Meta thesaurus that must be licensed since it includes proprietary content.

Bio Ontologies

- NCBO
- “BioPortal is an open repository of biomedical ontologies, a service that provides access to those ontologies, and a set of tools for working with them.”

National Center for Bio-Ontologies
Largely Medically related...

Bio-Ontology Example, the Foundational Model of Anatomy



Bio-Ontology Research Activities

- Develop the controlled vocabularies
 - Terms
 - Rules
- Schema Definition
 - Conceptual model
 - Implementation-specific schemas
- Domain maps are created
- Database query mechanisms are developed
- Resolution rules are defined
- More applications are developed to extend the utility and computational access of the resource

3/31/16

11

Ontology -- Review

- Philosophical study of:
 - Being
 - Becoming
 - Existence or reality
 - Categorization or being and their relations.
- Metaphysics.
 - What entities exist or may be said to exist.
 - Groupings, hierarchy
- Practical application in developing relationships

“Ontology is the philosophical study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations. Traditionally listed as a part of the major branch of philosophy known as metaphysics, ontology often deals with questions concerning what entities exist or may be said to exist, and how such entities may be grouped, related within a hierarchy, and subdivided according to similarities and differences. Although ontology as a philosophical enterprise is highly theoretical, it also has practical application in information science and technology, such as ontology engineering.” ~Wikipedia

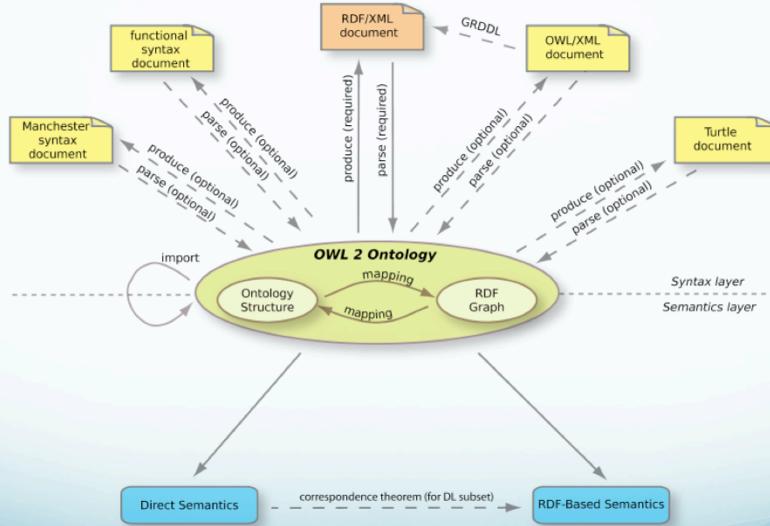
Onto ~ “Being, That which is”

OWL - DL

- W3C 2004 OWL Web Recommendation
- “The OWL Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications.”
- The OWL 2 Web Ontology Language
 - is an ontology language for the Semantic Web with formally defined meaning
 - provide:
 - classes, properties, individuals, and data values
 - stored as Semantic Web documents
- Written and exchanged in RDF

“The World Wide Web as it is currently constituted resembles a poorly mapped geography. Our insight into the documents and capabilities available are based on keyword searches, abetted by clever use of document connectivity and usage patterns. The sheer mass of this data is unmanageable without powerful tool support. In order to map this terrain more precisely, computational agents require machine-readable descriptions of the content and capabilities of Web accessible resources. These descriptions must be in addition to the human-readable versions of that information.”

QWL Design



- <http://www.w3.org/TR/owl2-overview/>

“The OWL 2 Structural Specification document defines the abstract structure of OWL 2 ontologies, but it does not define their meaning. The Direct Semantics [[OWL 2 Direct Semantics](#)] and the RDF-Based Semantics [[OWL 2 RDF-Based Semantics](#)] provide two alternative ways of assigning meaning to OWL 2 ontologies, with a correspondence theorem providing a link between the two. These two semantics are used by reasoners and other tools, e.g., to answer class consistency, subsumption and instance retrieval queries.”

OWL Property Restrictions - constraints

- There are value constraints and cardinality constraints on properties.
 - A property restriction on property P looks like the first example and a restriction based on NOT belonging to a group looks like the second.

```
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#P"/>
    <owl:allValuesFrom rdf:resource="#V"/>
  </owl:Restriction>
</rdfs:subClassOf>

<owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="CL:cell"/>
  <owl:Class>
    <owl:complementOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#CD8 receptor"/>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#has_part"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:complementOf>
  </owl:Class>
</owl:intersectionOf>
```

The class that contains this rdfs:subClassOf element is a subclass of an anonymous class of the form ‘all instances for which all values of property P are from class V’. This does not mean that an instance of the class must have the property P only that if it has P then it must have a value from V.

In the second example we define a class by stating what it is **not** – using the Cell Type Ontology CL, and we restrict the definition of cultured cells that do not express the CD8 receptor on the surface using the keyword owl:complementOf.

OWL and the Semantic Web

- OWL has 6 mechanisms for defining classes
 - Name: indicated by a URI
 - Enumeration: list all of the individuals
 - Property restriction: evaluate, when true the individual belongs
 - Intersection: of two or more class descriptions
 - Union: of two or more class descriptions
 - Complement: the not-included in a class description

```
<owl:Class>
  <owl:oneOf rdf:type="Collection">
    <owl:Thing rdf:about="#Adenine"/>
    <owl:Thing rdf:about="#Guanine"/>
    <owl:Thing rdf:about="#Cytosine"/>
    <owl:Thing rdf:about="#Thymidine"/>
  </owl:oneOf>
</owl:Class>
<owl:Class rdf:about="#A-Microtubule">
  <rdfs:label>A-Microtubule</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Cilium Microtubule"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#is Physical Part of"/>
      <owl:someValuesFrom rdf:resource="#Cytoskeleton"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#is Physical Part of"/>
      <owl:someValuesFrom rdf:resource="#Cilium"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The Web Ontology Language (OWL) was developed as a very generic language to enable interoperability of anything that can appear on the web. This required availability, structure rules, definition rules and consistency in applying the rules. OWL uses the Resource Description Framework (RDF) data model and language.

The logical model lets us reason about and extend existing relationships.

Definitions can be constructed using a keyword intersection_of also.

Classes are defined by placing restrictions on the things that belong to the class. Property restrictions are often used (second example).

Here an A-Microtubule is a complete cylindrical microtubule that is part of a microtubule doublet in a cilia.

A-Microtubule is a subclass of Cilium Microtubule

The two subClass of restrictions define that is must be a part of the Cytoskeleton and that is must be a physical part of the Cilium.

- The OBO ontology language models a subset of the semantics from OWL2 (the Web Ontology Language).
- Description logic (DL) is the reasoning model – has ‘constructors’ for : intersection, quantification, inverse roles and composition.
- An OBO document is a flat-file. Each line of the file header has a tag-value pair, separated by a colon.

```
format-version: 1.2
date: 13:03:2014 10:45
auto-generated-by: OBO-Edit 2.0
subsetdef: goslim_cerevisiae "Cerevisiae GO slim"
subsetdef: goslim_generic "Generic GO slim"
default-namespace: gene_ontology
```

- The sections of information after the header in a document are called *stanzas*
 - There are 3 keywords: universal types [Term], type definitions [Typedef] and instances [Instances].
 - A stanza begins with one of the 3 keywords and is followed by tag : value lines.

```
[Term]
id: GO:0000031
name:mannosylphosphate transferase activity
namespace: molecular function
def: "catalysis of the tranfer of a mannosylphosphate
group from one compound to another" [GOC:j1]
is_a: GO:0016740 ! transferase activity
```

1

Format-version is a tag (or an element) and its value is after the colon (here it is ‘1.2’) Auto-generated-by tells you what editor was used to create the file. This particular tool is widely used by the GO consortium, here we are looking at a GoSlim document describing GO terms for yeast.

Subsetdef tag tells you which dictionary (the cv terms) that will be used in the following information.

You can add empty lines for better human readability – parsers ignore them. However a NL or CR is interpreted as a new line, so you would not include that in real text.

The id is a unique identifier *within the dictionary* being used.

Each term has a name.

Within the default namespace you can have a sub-category.

In the Gene Ontology there are 3 related categories, Molecular Function, Biological Process and Cellular Component - the namespace in this part of the hierarchy tells which of these you are in.

The definition defines the function and the authority – here the authority is GOC, standing for the Gene Ontology Curator responsible for the definition – this gives provenance.

The type definition belongs to a controlled vocabulary within the ontology, here we have is_a: and a value, and a comment. Note that comments are noted with a ! (see

OBO ontology examples

This stanza shows the use of a synonym for a term, and references to other authorities.

```
[Term]
id: GO:0000121
name: glycerol-1-phosphate activity
namespace: molecular_function
def: "catalysis of the reaction: glycerol 1-phosphate + H2O = glycerol + phosphate." [EC:3.1.3.21]
synonym: "alpha-glycerol phosphatase activity" EXACT [EC:3.1.3.21]
xref: EC:3.1.3.21
xref: MetaCyc: GLYCEROL-1-PHOSPHATASE-RXN
is_a: GO:0016791 ! phosphatase activity
```

- Terms may have as many `is_a` (subclass) relations as you want.
- Any other type of relation is defined with a *relation type name* and a target term id, and it must be defined in a `Typedef` stanza – see below.

```
[Typedef]
id: part_of
name: part_of
xref: OBO_REL:part_of
is_transitive: true
```

```
[Term]
id: GO:0000136
name: alpha 1,6-mannosyltransferase complex
namespace: cellular_component
def: "A large multiprotein complex (more)" [GOC:mcc]
is_a: GO:0031501 ! mannosyltransferase complex
is_a: GO:0044431 ! Golgi apparatus part
relationship: part_of GO:0000137 ! Golgi cis cisterna
```

So for the `is_a` relationship type you do not have to declare the type (it is the default) but for any others you must declare it - here the `part_of` relationship type is cross-referenced to the OBO Relationship namespace, and a further characteristic is stated, which can be used for reasoning on new elements (that it is transitive, that is, if a is part of b and b is part of c then a is part of c is true).

Intersections – computable definitions.

- A reason to use a logical model is to be able to reason about and extend existing relationships.
 - Definitions are constructed using the keyword `intersection_of`.
 - If there are 2 or more intersections in a stanza then the term is equivalent to the *common intersection* of them all.

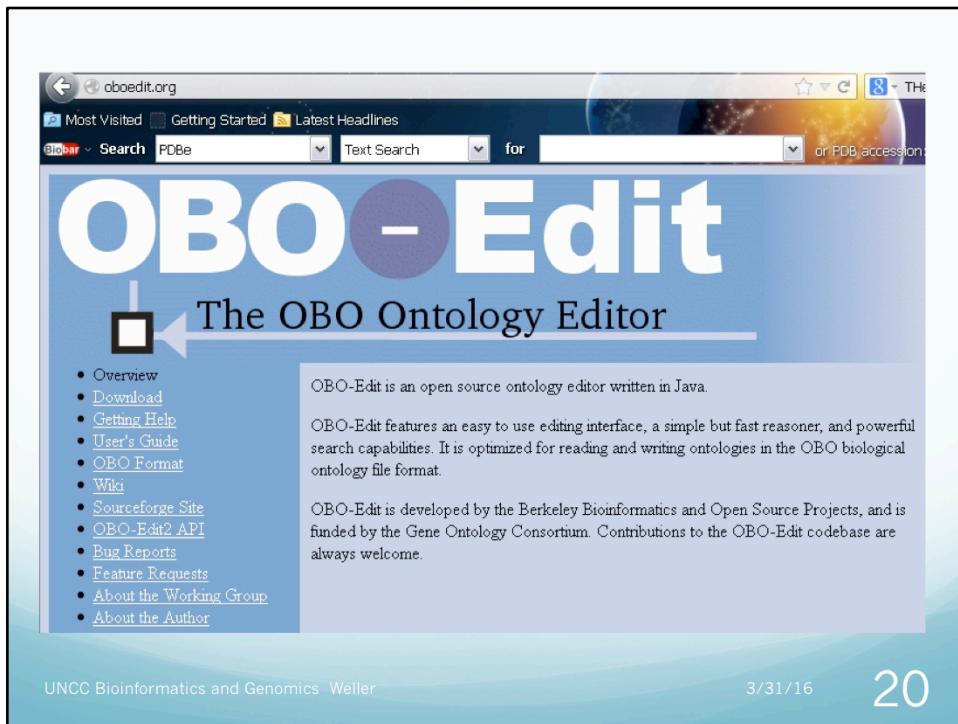
```
[Term]
id: GO:0000153
name: cytoplasmic ubiquitin ligase complex
namespace: cellular_component
def: "a ubiquitin ligase complex found in the cytoplasm" [GOC:mah]
is_a: GO:0000151 ! ubiquitin ligase complex
is_a: GO:004444 ! cytoplasmic part
intersection_of: GO:0000151 ! ubiquitin ligase complex
intesection_of: part of GO: 0005737 ! cytoplasm
```

- A computer ‘understands’ the logical definition the way a human understands the definition
- Declaring the equivalence of ubiquitin ligase complex that is in the cytoplasm and a cytoplasmic ubiquitin ligase complex is the result of a cross-product definition.
 - Cross-product definitions follow a genus-differentia pattern: the genus is the ubiquitin ligase complex and the differentia is the cytoplasm.

By computable we mean the logic is something that can be described to a computer = in this case a comparison is carried out that finds what is common between two sets, one a molecular process and the other a cellular component. The set is the collection of related items that your statements define.

Genus+ Differentia are the two parts of a definition. A genus is a family, which is an existing definition and a differentia is the portion of the new definition that is not provided the original definition. You are creating further subgroups, usually through a distinguishing characteristic and according to the accepted definitions of types.

Composing these two parts follows rules. Some of these are quite general – don’t create a circular definition, use a property that is an essential attribute, find a property that classifies a group of things and not a single individual, and declare a perperty in terms that are unambiguous. Ideally the property should be declared in a positive sense rather than in the does –not-have (negative) sense.



This began as GO-Edit, then DAG-Edit, then finally OBO-Edit.
It can output a number of file formats, including its own OBO format, and the GO format, and the OWL format.
A term is called a Class in this system – the entities and processes that you want to define and relate.
Relationship types are the allowed relationships (whose logic can be completely and accurately defined) such as `is_a` and `part_of` types – the ways that child classes descend from parent classes.
Relationships are the specific relationships that are assigned to a given parent and child term.
The OBO format was developed to track the large amount of meta-data that biological scientists had to manage, as well as the provenance that is essential to the scientific process. Some of the semantics are quite complicated compared to the Semantic Web as well.

Cardinality restrictions

- Cardinality can be constrained to an exact number, a minimum number and a maximum number using owl:cardinality, owl:minCardinality and owl:maxCardinality.

```
<owl:Class rdf:ID="Wine">
<rdfs:subClassOf rdf:resource="#food;PotableLiquid"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasMaker"/>
    <owl:cardinality>1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasMaker"/>
    <owl:allValuesFrom rdf:resource="#Winery"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:restriction>
    <owl:onProperty rdf:resource="#madeFromGrape">
      <owl:minCardinality>1</owl:minCardinality>
    </owl:onProperty>
  </owl:restriction>
</rdfs:subClassOf>
<owl:Class rdf:ID="WhiteBurgundy">
  <owlintersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Burgundy"/>
    <owl:Class rdf:about="#WhiteWine"/>
  </owlintersectionOf>
</owl:Class>
<owl:Class rdf:about="#WhiteBurgundy">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:hasValue rdf:resource="#ChardonnayGrape"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:maxCardinality>1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

This is a class definition for a wine varietal

It is an intersection of classes White Wine (a subclass of wine) and Burgundy (a region and a style of wine that is made from exactly one grape variety, Chardonnay)

Because we made a constraint on madeFromGrape to have a minimum cardinality of 1 inherited from the class Wine, and a maximum cardinality of 1 for the class White Burgundy, this type of wine can be made from exactly 1 varietal of grape.

OWL Property Restrictions and functions

- In addition to IntersectionOf, OWL has a UnionOf operator that lets you *combine* 2 collections
- Property characteristics are defined so that logical rules can be evaluated
 - Owl:Transitive Property (Mitochondria in Cytoplasm and Cytoplasm in Cell so Mitochondrion in Cell)
 - owl:SymmetricProperty (hasSameColorAs) expresses equivalence
 - owl:inverseOf lets you express if an edge has two arrows (has_part and part_of)
 - Owl:disjointWith (for classes). Use differentFrom (for two individuals), distinctMembers (for two sets of individuals)
- You can define a function with owl:functionalProperty as the example shows.

```
<owl:ObjectProperty rdf:ID="encodes Amino Acid">
    <rdf:type rdf:resource="#owl:FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Codon"/>
    <rdfs:range rdf:resource="#Amino Acid"/>
</owl:ObjectProperty>
```

The function declares that an amino acid is always coded for by a specific codon. The inverse is not true since the code is redundant.

Editing OWL with Protégé

- Protégé is an editor application used to build ontologies that has a graphical user interface and can display ontologies in a number of ways. It is free and open source, and frequently updated.
 - <http://protege.stanford.edu>
- For a tutorial on getting started, see the University of Manchester Web site
 - <http://owl.cs.manchester.ac.uk>
- Most OWL ontologies are developed with Protégé.

Summary

- The OBO ontologies have been designed to support science, especially the biomedical sciences – the emphasis has been on use of the ontology.
 - Most are taxonomies, classifications of a domain of biology using `is_a` and `part_of` compositional relations for the most part.
 - OBO has been extending its syntax to include structures like the intersections.
 - OBO is a significant part of the OWL-DL (data definition language)
- OWL ontologies have been developed by computer scientists who have focused on the frameworks, algorithms and software
 - The RDF model and RDF Schema language are the basis of the OWL framework.
 - There is ongoing work to be able to readily interconvert between OWL and OBO.