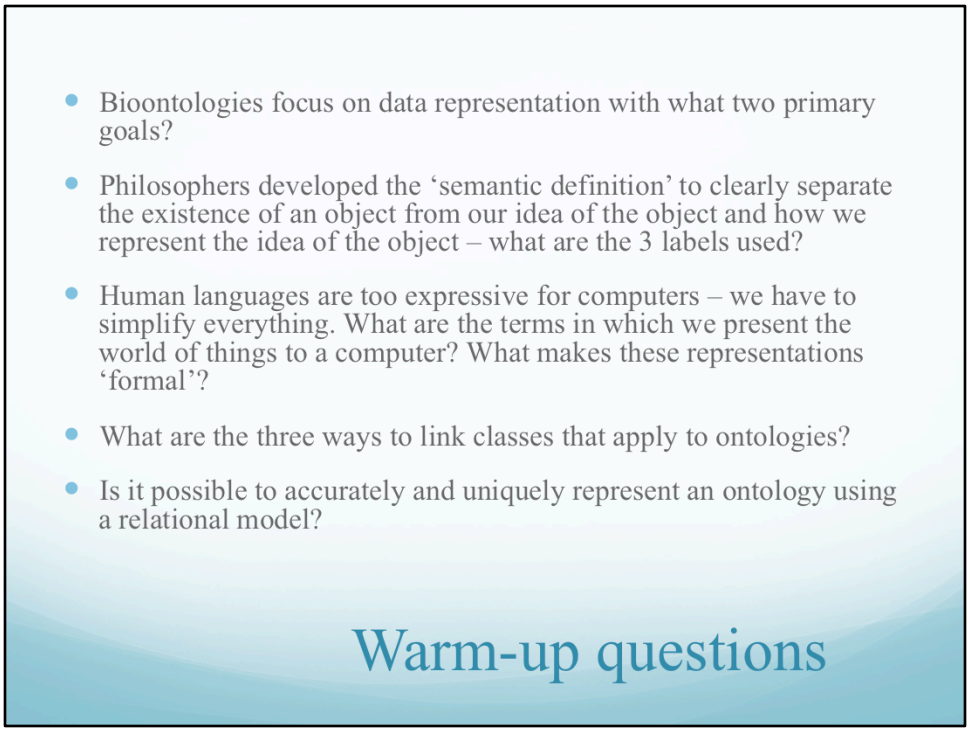
BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #16

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

Review Relational Model



Data exchange and data mining – you have to be manipulating things of the same type or with the same relationship to each other before a pattern can be meaningful, or extending the pattern to new things can be meaningful. The goal is to correctly predict features and relationships, to group items by features and relationships and to identify correlations from shared properties and trends.

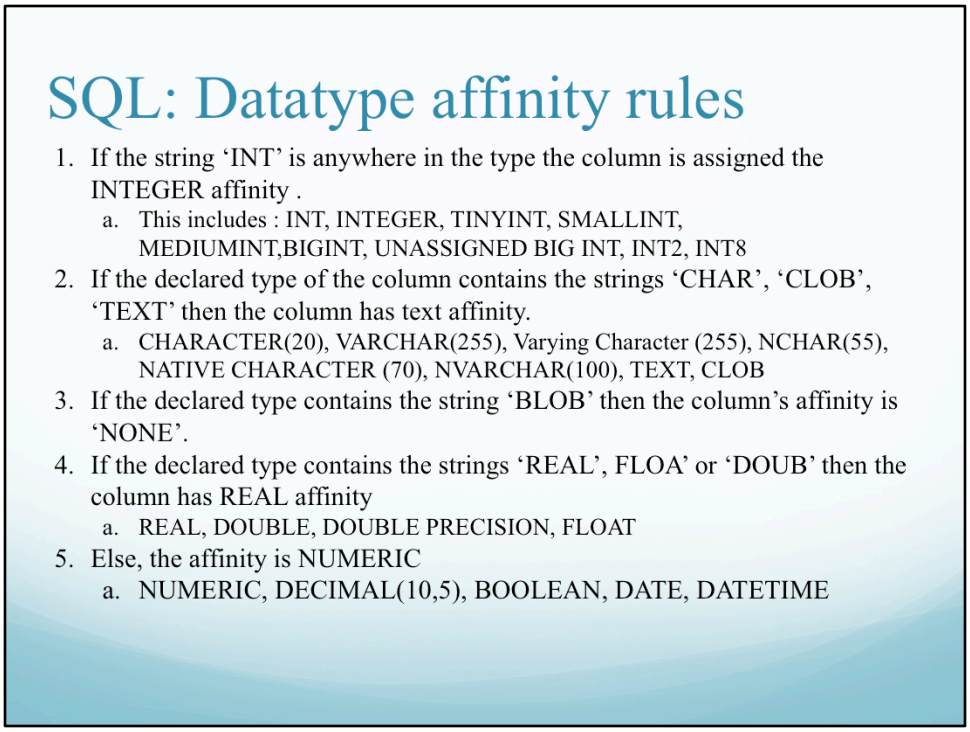
The thing itself, which has physical properties in time and space, the concept of the thing which we build in memory based on symbols for sensory data which stand for the thing and evoke the concept – so we might never have experienced the thing but someone else's symbols may evoke it sufficiently that we 'recognize' it when we see it.

Computers never have sensory experiences so the focus is on concepts and the properties used to evoke them, and the relationships that link them. The concepts are called units, which stand for classes and inidividuals whose relationship (the predicates that state facts about the units) can be stated using a logical structure, and the properties are called 'frames'.

A hierarchical structure allows only a single parent (many children). So to walk across the graph you have to go up to a common node and back down. It is possible to enumerate all possible paths through the tree structure, so you can do a

SQL Topics

- Data Affinity
- Casting Types
- CASE Statement



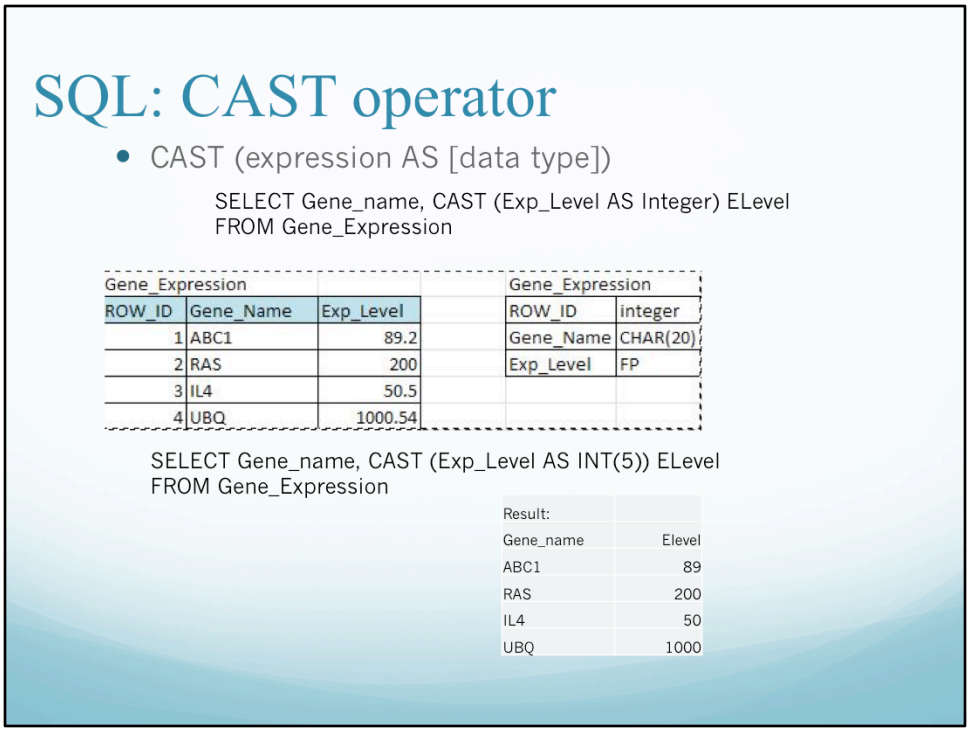
Most SQL database applications use a static, rigidly defined typing that sets the allowed set of values and their interpretation by the definition set up for the container (the attribute field).

A dynamic typing system keeps an interpretation that is associated with the value itself, not its container. To communicate this difference, SQLite describes storage classes rather than datatypes. We had 5 storage classes (NULL, integer, real, text and blob).

To stay somewhat compatible with other SQL databases, SQLite does have a type affinity definition for columns – the recommended, but not required, data to be stored in the column.

The affinity classes are TEXT, NUMERIC, INTEGER, REAL and NONE. The storage classes are described with rules such as:

TEXT affinity can store data using storage classes NULL, TEXT and BLOB.
NUMERIC affinity may contain values using all five storage classes.If the data is text and only contains numerals then the storage class is converted to INTEGER or REAL if the conversion is lossless and reversible. If that conversion is not lossless and reversible then the value is stored using the TEXT storage class.
You can then have subclasses - INTEGER has 6 subclasses, allowing for different



The CAST expression is used to convert the value of <expr> to a different storage class than the original data type has (the type-name).

In the database you can make some column type conversions, so long as not information is lost. With the CAST operator you can force a conversion, even if you do lose information.

If the value of <expr> is NULL then the result of the CAST expression is also NULL. The other conversions follow the affinity rules.

When I start with REAL number and CAST into an integer gives you the integer that is between the REAL value and zero that is closest to the real value.

In the example on the slide, we cast from a floating point to a character (string or text). The float values were truncated rather than rounded – read the documentation on your system to make sure you know how that will be handled.

SQL: CAST Conversion in SQLite

Affinity of <type-name>	Conversion Processing
NONE	Casting to NONE converts the value to a BLOB, first casting it to TEXT in whatever encoding scheme is used.
TEXT	BLOB --> TEXT interprets the sequence of bytes according to the declared encoding scheme. INT or REAL --> TEXT
REAL	BOLB --> REAL goes through TEXT as above. TEXT --> REAL takes the longest possible prefix of the value that can be interpreted as a real number. Leading spaces in TEXT are ignored. If there is no prefix the conversion yields 0.0
INTEGER	BLOB --> INTEGER through TEXT. TEXT --> INTEGER as above except the result of conversion in the absence of a usable prefix is 0. REAL --> INTEGER results in the integer between REAL and zero that is closest to the REAL value. If the value of the largest signed possible integer is exceeded then the largest available is used.
NUMERIC	TEXT and BLOB --> NUMERIC forces conversion to REAL and then to INTEGER only if this is a lossless reversible conversion. You cannot cast REAL or INTEGER to NUMERIC.

There are examples of conversions on
https://www.sqlite.org/datatype3.html#storageclasses

SQL: CASE function

- A CASE expression gives you the logic of an IF-THEN-ELSE loop in other programming languages.
  - It is placed within a SELECT statement's target list
  - It terminates with END

```
CASE [optional exp1]
WHEN [condition 1 returns Boolean value] THEN [result1]
WHEN [condition 2 returns Boolean value]
[….]
ELSE [optional default result]
END  [AS alias]
```

  - The optional expression 1 is called the base expression – it is evaluated just once and that result is compared against the evaluation of each WHEN expression (left to right).
  - The result of the CASE expression is the evaluation of the THEN expression that corresponds to the first WHEN expression that is TRUE, or if none are true then the ELSE default result is returned. If there is no ELSE included and no WHEN expression is TRUE then a NULL is returned.
  - ELSE can include a subquery (Select…)

CASE example

```
CREATE TABLE CITY_STATE (rowid  INTEGER PRIMARY KEY NOT NULL, city TEXT, state TEXT);

INSERT INTO CITY_STATE (rowid, city, state) VALUES (100, 'Springfield', 'MA');
INSERT INTO CITY_STATE (rowid, city, state) VALUES (101, 'Washington', 'DC');
INSERT INTO CITY_STATE (rowid, city, state) VALUES (102, 'Mobile', 'AL');
INSERT INTO CITY_STATE (rowid, city, state) VALUES (103, 'Wilmington', 'DE');
INSERT INTO CITY_STATE (rowid, city, state) VALUES (104, 'Roanoke', 'VA');
INSERT INTO CITY_STATE (rowid, city, state) VALUES (105, 'Seattle', 'WA');

sqlite> SELECT state, city || ', '|| CASE
...>    WHEN state='AL' THEN 'Alabama'
...>    WEHN state='DC' THEN 'District of Columbia'
...>    WHEN state='DE' THEN 'Delaware'
...>    WHEN state='MT' THEN 'Montana'
...>    WHEN state='MA' THEN 'Massachusetts'
...>    END AS 'city and state'
...>    FROM CITY_STATE;

state|city and state
MA|Springfield, Massachusetts
DE|Wilmington, Delaware
DC|Washington, District of Columbia
AL|Mobile, Alabama
sqlite>
```

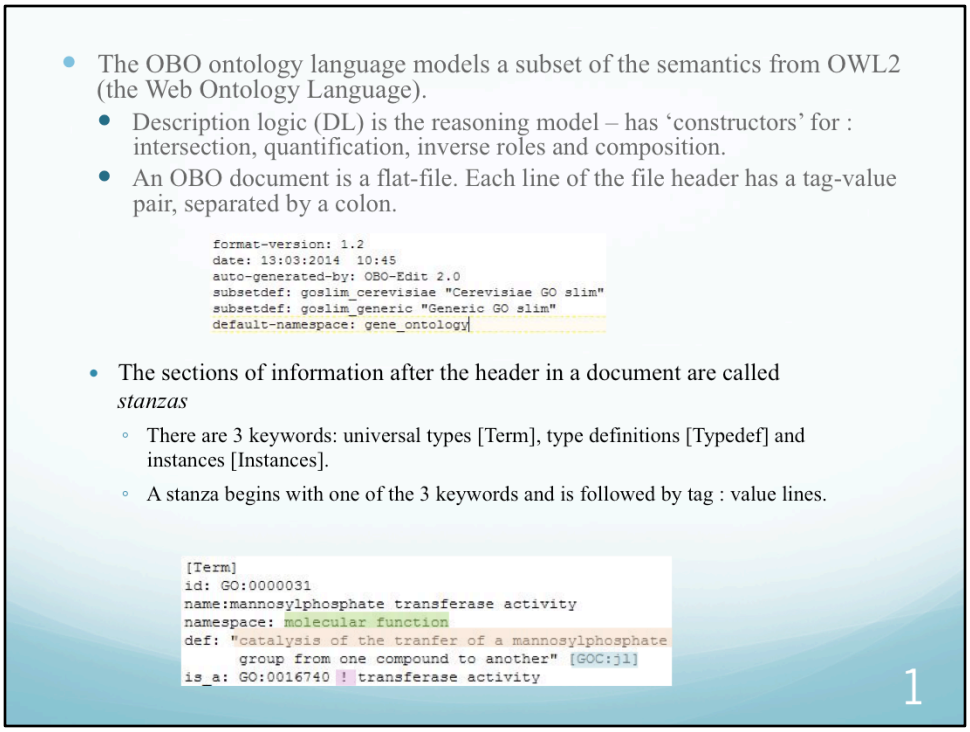
See http://sqlite.awardspace.info/syntax/sqlitepg09.htm for more (and more complicated) query examples that use subqueries and aggregate functions

The ‘MT’ evaluates to FALSE, so there is no Montana output. And there was no evaulation of Washington (state) or Virginia, so those did not get evaluated and put in the output.

You could also use this to output only gene expression values greater than some cutoff.

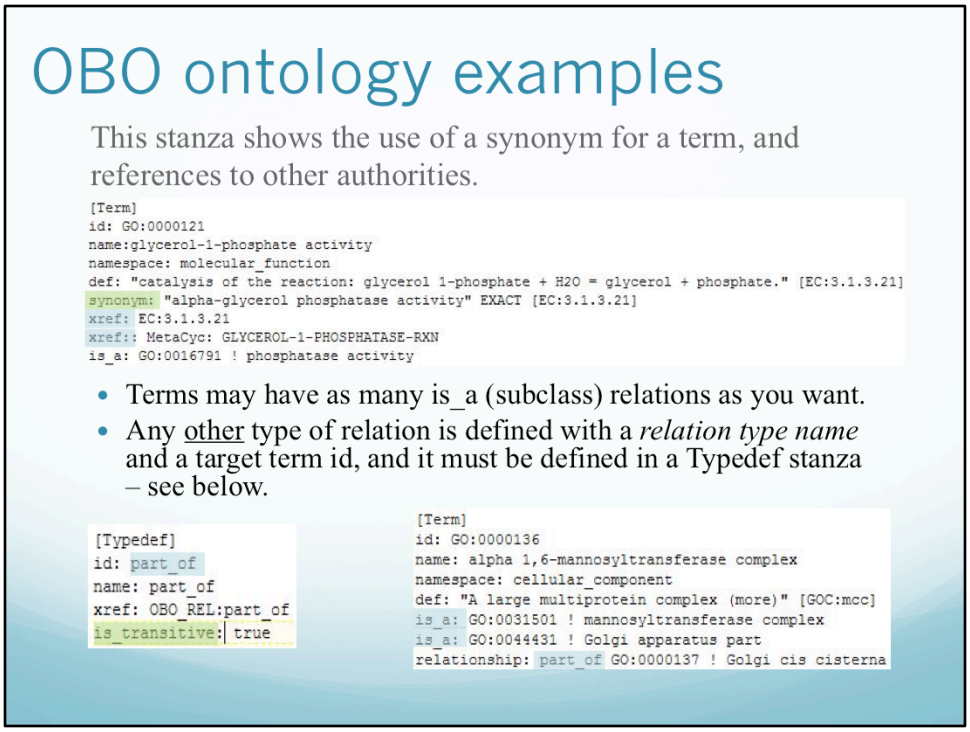
- OBO and OWL
- The Gene Ontology
- The Sequence Ontology
  - The Genbank Flatfile 3 (GFF3) format

Topics (extended)

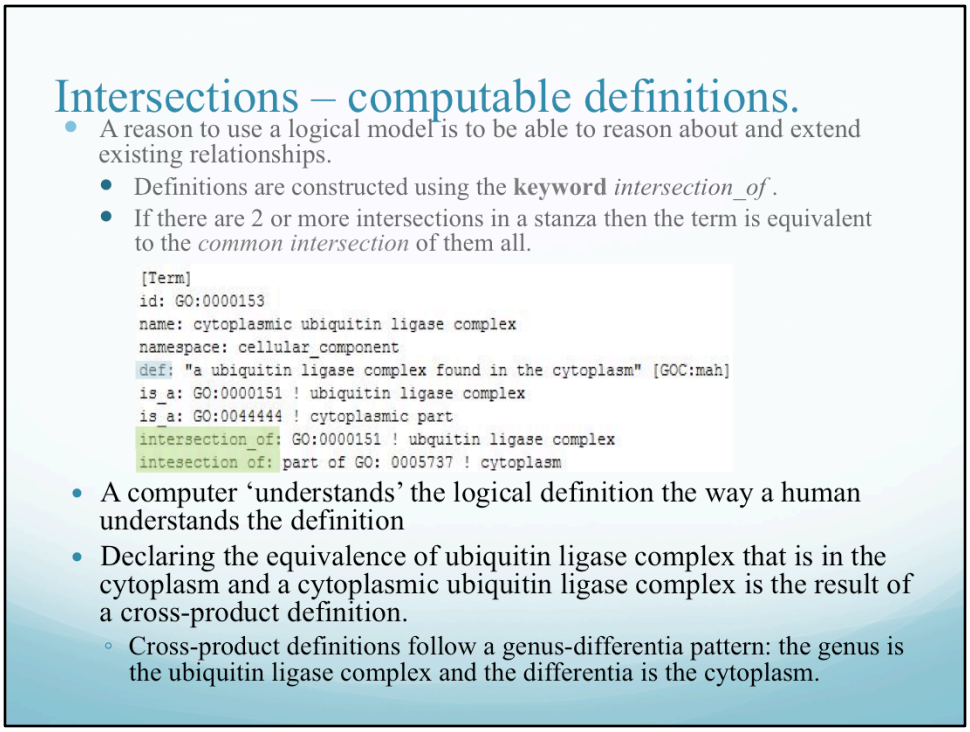


Format-version is a tag (or an element) and it's value is after the colon (here it is '1.2')
Auto-generated-by tells you what editor was used to create the file. This particular tool is widely used by the GO consortium, here we are looking at a GoSlim document describing GO terms for yeast.
Subsetdef 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 s controlled vocabulary within the ontology, here we have is_a: and a value, and a comment. Note that comments are noted with a ! (see



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).



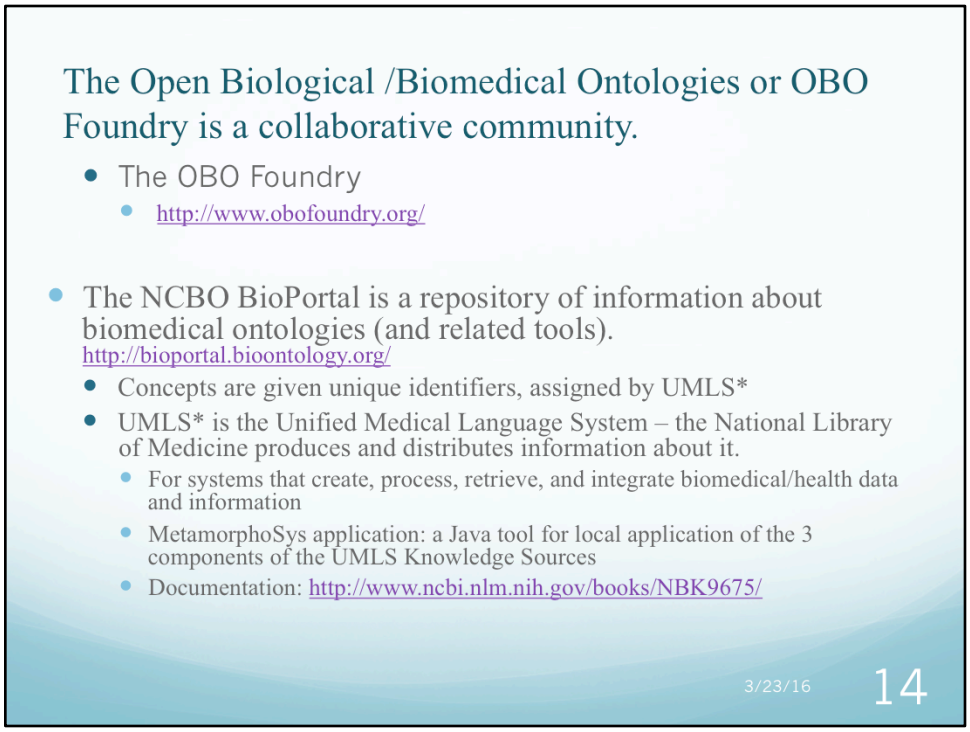
By computable we mean the logic is something that can be described to a computer = in this case a comparison is carried out that finds what is common between two sets, one a molecular process and the other a cellular component. The set is the collection of related items that your statements define.

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.

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 OW-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.



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.