Module 1 Readings

Introduction

This week we'll review databases and get started on various database issues. As we visit specific databases during the course, look for how the databases do, or do not, address various issues that you read about.

Regarding the specific databases we visit... There are thousands of biological databases — obviously we cannot discuss each one in a semester. Therefore, I've chosen some of the "big name" resources that are highly useful and that demonstrate many of the principles we'll cover. In addition, we will touch on some less well known databases that also are good examples. Finally, for your own course project, you will need to review related work — this will be your opportunity to explore any databases that are of particular interest to you.

We'll also look at how information is obtained from databases. Methods include:

- text search — the province of Information Retrieval (IR)
- SQL
- browsing/navigation
- programmatic (such as web services)
- BLAST
- FTP (downloading)

Topics in the Lecture Content:

- Database Review, including Essential Steps of the Database Lifecycle
- Database Review of Database Types
- Data Models and Data Modeling
- Accession Numbers
- Database Issues – Data and Database Integration
- Information Retrieval Theory Overview
- Overview of NCBI Resources
- Additional Databases

When looking at a resource, pay particular attention to:

- What information is contained in this database or resource and what is its data model (when given)?
- What types of questions can be answered with this information?
- How is the information searched/accessed and how is it presented to users?
- How is the data interconnected with data in other databases or resources?
- How does the access/presentation/interconnection help (hinder) the ability of users to answer their questions?

Database Review

Basic terminology

- data abstraction: a more general representation of information. Data abstractions can be hierarchical with each level being more general than those below it. Primarily, the abstraction must be useful for a purpose — it makes understanding a problem or framing questions easier for a particular problem area. For example, a database of protein interactions may present a high level of abstraction that just shows the links between interacting molecules; a lower level abstraction may provide details about the physical contacts between molecules (specific amino acids, duration of interaction, etc.).

- database: a collection of related information (data) and associated data dictionary.
- transaction: one or more actions that reads and/or changes the database. May be done by a human and/or a program.
- data dictionary: a description of the data itself. This is also called meta-data or the system catalog. It includes definitions and representations of the data elements and the database structure.
- entity: a set of objects having the same properties, and that are independent and important in the problem representation or solution.
- entity occurrence: a particular (unique) occurrence of an entity (object).
- relationship: a set of meaningful connections or associations among two or more entities. (Sometimes relationships are self-referential.)
- relationship instance: a particular (unique) association between two entities.
- model: a representation (at whatever level of detail is appropriate for the problem being addressed) of real world things and events.
- **metamodel:** the language used to express a data model.
- **data model**: an abstract description of the representation and use of data, including the relationships between things and any constraints. Note that this term has different meanings in different textbooks and papers. Sometimes the terms **schema** or **meta-model** are also used. There are two typical senses of the word data model (or schema and meta-model):
    1. data model theory: a definition of the type of data structures, operations, and rules that are used to describe the data — basically the language used to express the data model (for example, the relational model).
    2. data model instance: what is produced when a particular data model theory is used to describe some real world data. This is can also be called a logical data model.

  Sometimes textbooks make a distinction between schema and data model — often in this situation, schema is a data model instance and data model is data model theory. You may use whatever terminology you like or are familiar with, just be aware that there are difference uses of these terms out there.

- physical data model: a data model that is tied closely to how the data will be physically stored and the particulars of the database management system used to contain the data.
- relational (database) model: a way of describing data and relationships (a metamodel) using the mathematical concept of "relation". In a particular instance, the data is represented in one or more tables.
- attribute: in the relational database model, a piece of information that describes something about the relation (that is, a column name in a table). Attributes are also used in the object model and describe something about the object. We can also say that entities have attributes.
- arity or dimension: in the relational database model, the number of attributes of a relation (table).
- scheme: in the relational database model, the name of the relation (table) plus the names of the attributes (columns) of the relation.
- tuple: in the relational database model, each row is called a tuple.
- database scheme: in the relational database model, all the relations (tables) with their attributes (columns). (This could also be called the data model or schema or meta-model.)
- entity-relationship modeling: describing data via entities and relationships (metamodel). Various pictorial conventions are used to represent entities (sometimes boxes) and relationships (sometimes arrows between boxes).

- **object-oriented model** – classes have attributes and methods (functions); subclasses inherit from parent classes and extend the model with (possibly) additional attributes and methods. An object is an instance of a class.

Note that there is some difference of opinion regarding what information to include in a data model. I am of the opinion that if you are using a relational metamodel, you should include table names and column names, as well as data types. However, I would allow that the names and types could be more descriptive than what is actually used in the physical model (or what is actually implemented). For example, you may have a table named “Protein Protein Interaction” in the logical data model but actually call the table PPI when you create the database. Furthermore, there can be differences of opinion over what should be in a logical versus a physical model. There is also something called a conceptual model, which is a very high-level representation of the information, above the logical level. If you are working with a team of people, there may be a definition already used that indicates what goes where. If not, reaching a consensus is best. If you are working on your own, do what you feel makes the most sense given the information and the end users. You will see various data models and will get a chance to do data modeling in the course to gain experience with this.

Essentials of the Database Lifecycle:

1. Planning: define the high-level objectives of the database system.
2. System definition: determine scope and describe what the database must do from the perspective of users and their activities.
3. Define requirements.
4. Logical database design: describe — independently of any particular database management system — important entities and relationships that must be in the database.
5. Select a database management system.
6. Physical database design: determine how the logical design will be implemented in a particular database management system.
7. Application design: design actions performed by users or programs, design the user interface, and ensure all requirements are being met.
8. Implementation: develop the actual database, including data conversion and loading, and application programs. May include a prototype step.
9. Test the system.
10. Maintenance and improvements.

There are a number of variations of the above, including "rapid prototyping" (where a database system is quickly built for the purpose of defining requirements and testing design ideas, with the intention of the prototype being tossed away) and "spiral development" (where various steps are formally planned to be repeated), but the above activities will take place in some fashion or another when developing a database. You will do steps 1-9 in a simplified form for your course project.

Database Review of Database Types

Please insure that you take the time to review basic database principles on your own as needed. A few important points will be brought up here.

Flat Files. A flat file "database" is a large collection of files conforming to a standard format. Sometimes the standard is truly a standard recognized by a standards organization, Sometimes the standard is standardized for the application using the files.

A flat file "database" is analogous to a large collection of papers put into a file cabinet and ordered according to a field label or title. Many bioinformatics databases began (and often still exist) as flat file databases. There is nothing necessarily wrong about flat file databases. With proper indexing, they can be made completely searchable. However, a disadvantage of using flat files is that you (the programmer) need to provide code for accessing, updating, managing transactions, or otherwise using the database — functionality that is provided by default from a relational or object-oriented (or other NoSQL) database management system.

That said, there are flat file "databases" that provide the functionality for managing the files and data, alleviating at least some of the burden on the programmer. For example, there are many XML-related technologies for working with data in XML files. (We introduce XML (and JSON, another format) in the next module.)

We will see below that the Gene Ontology uses a flat file format (OBO, for one).

Sometimes a flat file format is used within a relational database. GenBank (see later in the reading) is one such database that uses a flat file format, the standard ASN.1. The information in a GenBank record is formatted using ASN.1 *internally*, and is kept in a relational database blob in what is called the ID management database.

In the case of GenBank, there is another "flat file" format, namely the format used to *output* the GenBank record that users see when browsing GenBank. This is the "GenBank flatfile format" often referred to in textbooks. We will see the flat file format in more detail below.

E-R Model. You should know the basics of E-R Models — review the concept of entity-relationship modeling as needed. This is a technique for modeling information and relationships that will be contained in a database. You will need to describe/diagram the data model of your course database (and for one of the homeworks). You may want to use E-R diagrams to do this. View http://www.umsl.edu/~sauterv/analysis/er/er_intro.html

Another good link is http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model. View the diagram on different styles for representing relationships and the section on Crow's Foot notation, which is very commonly used. You may wish to use one of these styles when documenting your data model.

Relational Databases. Relational database technology has been around for decades and is well established. Complex, time-sensitive operations and data have been deployed for many years. SQL is a typical method for accessing information in a relational database. Here is a very quick overview of SQL.

Here is an SQL statement which retrieves names of employees in the department whose ID is CS:

```
SELECT Employees.Name
FROM Employees
WHERE Employees.DeptID = "CS"
```

We will begin by trying to understand this SQL statement. To understand any SQL statement, it is, of course, important to consider the FROM clause.

```
FROM Employees
```

means that the data retrieved will originate from the Employees table.

Next, the WHERE clause.

```
WHERE Employees.DeptID = "CS"
```

means that the data retrieved will come from rows whose DeptID column has the value CS.

And the data to obtain is indicated by the SELECT clause (Employees.Name).

Another important concept is JOIN (in particular INNER JOIN). As a reminder, here is a very simple example. Let's say you have a table with gene ids and their associated protein id (ignoring any issues with isoforms, multiple gene names for the same gene, etc.). And there is a second table that gives protein-protein associations. Suppose the two tables have the following records:

Table GeneProt			**Table ProtProt**	
Column geneId	**Column protId**		**Column prot1Id**	**Column prot2Id**
gene1	prot1		prot1	protA
gene2	prot2		prot3	protC
gene3	prot3		prot5	protE

This next command will obtain the gene ids for those proteins in the prot1Id column that have a gene id in the GeneProt table. (By default in Sqlite, which we will be using in Homework 2 and 3, "join" is an inner join.)

```
SELECT geneId FROM GeneProt JOIN ProtProt WHERE GeneProt.protId=ProtProt.prot1Id;
```

```
gene1
gene3
```

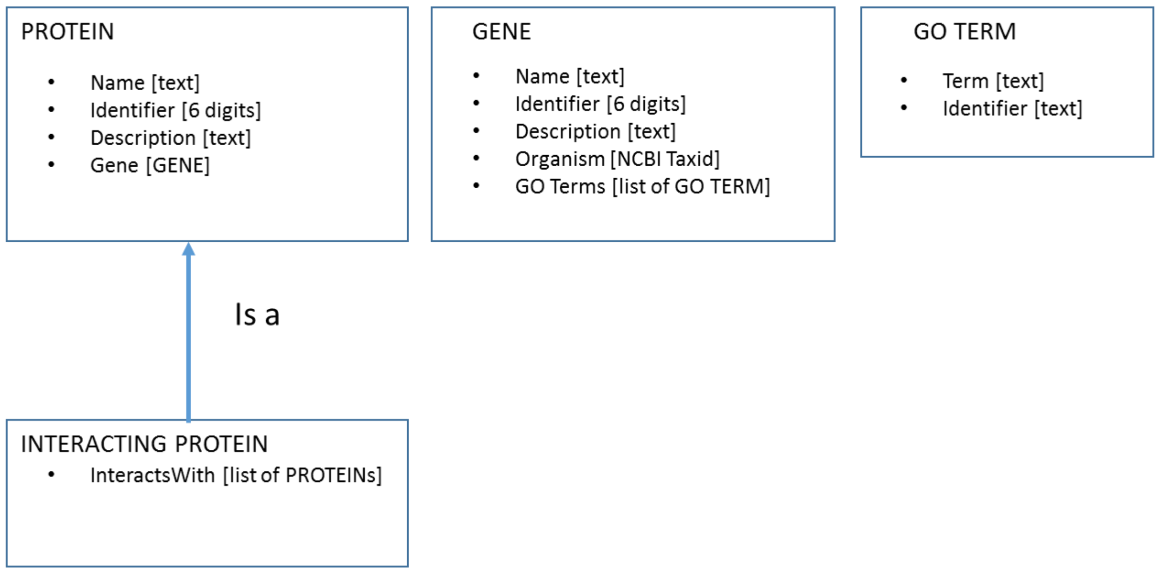
You should feel comfortable with commands for creating tables and inserting data (could you recreate the two simple tables above?) and selecting data, along with doing inner joins.

If you need it, an SQL tutorial can be found here: http://www.sqlcourse.com/ and http://www.sqlcourse2.com/. Review this as you feel you need to.

Object Databases. Object Databases are an example of a database where the model for the information can be used to store the information in the database as well as manipulate it programmatically without the need to convert to another model (such as the relational model). They are particularly useful when developing in an object-oriented programming language such as Java or C++. The following quote is from a web page that is unfortunately no longer on the web, but it has a good concise definition.

"An object-oriented database system must satisfy two criteria: it should be a DBMS, and it should be an object-oriented system, i.e., to the extent possible, it should be consistent with the current crop of object-oriented programming languages. The first criterion translates into five features: persistence, secondary storage management, concurrency, recovery and an ad hoc query facility. The second one translates into eight features: complex objects, object identity, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility and computational completeness."

A simple object definition for a protein and its associated gene is the following:



In the above, the lists can be thought of as indicating a relationship with the object type (or class) for the elements in the list. Likewise with a type such as GENE. INTERACTING PROTEIN is a subclass of PROTEIN and inherits the attributes of PROTEIN. The above is simple diagram in The Unified Modeling Language (UML), which is a metamodel for expressing objects (and many other things) that we will not cover here (but is something you can use, should you decide to work with objects for your class project).

Note: Object databases can be considered a type of "NoSQL" database. We will look at some different kinds later in the course, but I want to point out now that a database doesn't have to be relational and to say that if you are interested in working with a NoSQL database for your project, you can.

Object Relational Mapping (ORM)

The above simple example could be stored in an object database or it could be easily converted to a relational model and stored in a relational database. Given the stability and excellent performance of relational database technology, storing object data in a relational database can be an appropriate choice. With an object-relational mapping library (for writing software), once the details of how to do the mapping are worked out, the library will allow accessing and storing elements in the "object" database thinking of the entities as objects and not as rows in a table.

In this example, we can define a table for PROTEIN, GENE, and GO TERM with columns that correspond to Name, Identifier, etc. Given that the identifiers are unique and are keys in the tables, we can use the GENE Identifier in the PROTEIN table to link the appropriate entities in a manner similar to GeneProt and ProtProt is the SQL example.

A list of entities can be handled with a separate table that links these entities together. The ProtProt table above already does this. If prot1 also interacts with protF, we would have another row with the tuple (prot1, protF) in the table.

In the above object example, GENE has a list of GO TERM (but there is no association from GO TERM back to GENE). A separate table again will handle linking these items:

Column GeneId	Column GOId
123456	GO0000824
123456	GO0099522
111111	GO0005840
222222	GO1901193

Data Models and Data Modeling

In the simple example we just looked at, what is the data model? Is it an object model or a relational model? I would consider the first variation to be object-oriented and the second to be relational. But the information contained is the same. Thus, for this course, we are going to consider a data model as the things/entities with their attributes and relationships of the information we are interested in working with to meet some users' needs or questions. We won't be overly concerned about the model's type.

Still, we need a way to write down our data model. To do that we need a language – a metamodel. Our language could be all text based and descriptive. It could be diagrammatic, it could be tabular. In the simple example, we could choose to use an object-oriented metamodel and express it using UML or some other diagrammatic method (it should allow us to provide the names and data types of all entities, attributes, and relationships, so it could be accompanied by a data dictionary).

We could choose to use a relational metamodel instead, in which case we would use tables and columns to write down our model (also giving the types of the columns such as integer and indicating which are the keys).

But in both cases the entities, attributes, and relationships of the data model are the same. The only difference is the need for linking tables in the relational version.

So to expand -- for this course, we are going to consider a (logical) data model as the entities with their attributes and relationships of the information we are interested in working with without worrying too much about the metamodel used to express the data model.

And, as you've seen above with Object Relational Mapping, you could define a data model using an object metamodel but use a relational model to store the data. Consequently, in this example, we will consider the **physical data model** to be a relational data model.

To further complicate matters, we will often be more interested in what data is presented to the user – what the data model is of the information you are looking at. This data model could be a subset of the data model for the entire database. Even if all data in the database data model is being presented, it could be organized in a way that is very different from the logical data model and especially the physical data model. We briefly saw above, and will see again below, that GenBank uses a relational database to store data, the ASN.1 format as the physical data model for the data, but uses the GenBank Flat File format to present information to the user.

Data modeling is looking at information that you want to store, access, and manipulate, and deciding what the important things/entities and relationships between those things/entities are. It includes defining attributes for the things/entities and defining types of relationships (such as "is a" or "part of"). Sometimes it can be difficult to decide what constitutes an important thing or entity and what the necessary attributes and relationships are. And two different people can come up with different data models for the same problem (although hopefully not too different).

Accession Numbers

You will notice various numbers when looking at sequence records when we visit GenBank (and other databases). Accession numbers uniquely identify records. These identifiers are quite important in referencing sequences and other data that have been stored in major databases. And, there might be several accession numbers associated with a biological molecule.

This multiplicity of identifiers can be an issue with biological data. In addition, as we'll see, what constitutes a unit of information that is given an identifier can be difficult to decide upon.

Many databases, like GenBank which we'll visit below, use an accession number + version number to uniquely refer to a record. The database/resource should publish what types of modifications cause a change to a version versus an accession number. In GenBank, if the sequence itself is modified, this causes a new version number. Changes to the record (the description of the sequence – that is, the metadata) do not cause a change; however, the modification date of the record will change. (https://www.ncbi.nlm.nih.gov/Class/MLACourse/Modules/Format/exercises/qa_accession_vs_gi.html)

Some example accession numbers: U49845, DQ057997.1, NM_001031630.1

Database Issues

In addition to practical experience with particular biological databases, we are also going to look at general database issues. As we go along, we will see how these issues come into play with the various databases we examine in detail.

We'll look at several issue "themes" over the rest of the course.

1. Handling very large amounts of data
2. Interoperability of databases — the integration of data from multiple sources to address a research question or enable knowledge discovery. Standards are often used here.
3. Information trustworthiness — data provenance and evidence
4. Presenting information to enable answering questions or knowledge discovery
5. Handling variations in or different levels or granularities of information
6. Dealing with the names of things and/or determining how a particular unit of information is to be represented in the database

Data and Database Integration

Let's begin by considering some aspects of integrating data and database integration. Bioinformatics is defined by NIH as "research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral, or health data, including those to acquire, store, organize, analyze, or visualize such data." As you can imagine, this definition encompasses a wide variety of databases and tools.

This diversity has led to a problem — often data from several sources must be integrated to provide a complete answer. That is, the sources must integrate and interoperate at some level — the level necessary to address the question being posed. Often the integration is complicated because of the different syntax (structure) of data, and is even more difficult due to the different semantics (meanings). These differences require the following actions to provide integration between diverse resources:

- Data transformation involves mapping data in different formats to some common structure (perhaps permanently or on-the-fly).
- Concept identification and resolution involves determining whether or not data in different sources refer to the same thing or not, and resolving conflicts between data.

  Sometimes the distinction between meanings of a concept may be subtle. For example, one source may contain sequences for only known genes while another includes sequences for postulated genes. What is the meaning of the word gene in both cases? Is it the same? From a "computer science" perspective they might be seen as the same (both may be an object or data structure holding a string with combinations of AGCT). But the meaning is different, and it may be very important to a biologist researcher to only get sequences from known genes and not from postulated ones. If it is an important distinction then that distinction must be represented somehow and utilized appropriately if the two data sources are integrated.

Or, there may be a situation where two resources have the same meaning for gene (for example, both include predicted genes), but the metadata associated with a gene may differ. One source may provide information about how the gene prediction was made and the other may not. If one, combined resource is made from both, some gene entries in the combined resource will be missing this information. If the combined resource schema requires this data, any gene missing this data will be considered invalid.

Or, the situation may be that both resources involve predicted genes, and both give information about how the prediction occurred — but they used different prediction methods. This may actually be a good thing if the two predictions give different evidence supporting the same conclusion. But if the genes predicted are not exactly the same, it may be difficult to combine the two gene records into a common resource without human intervention (or even with). This could occur, for example, because two different gene predicting programs may not identify the same exons. They may have some of the same exons but each may have one or more exons not predicted by the other. And/or one or more exons may have different starting and ending positions. In this type of situation, the data integrator needs to decide what information is important to obtain from both sources, whether to merge information or maintain separate data items from both, or to choose one source's information over the other's. The decisions made will be reflected in the data model of the integrated information and will determine what type of questions a researcher can answer when using the integrated resource.

Semantics are further complicated by the fact that there is no consistent, monolithic set of meanings applied across all bioinformatics systems. For example, even genes can have different names (although the HUGO Gene Nomenclature Committee unifies human gene names, see the end of this module). This diversity of meaning has arisen due to the fact that many bioinformatics resources were developed by individual research communities that each has their own way of looking at, and describing, problems and solutions.

Other issues to note:

1. A very large number of resources exist that have a high degree of heterogeneity — extremely diverse in format, scope, semantics, access methods.
2. These resources are also autonomous. They change their content, data model, user interface, etc., without coordination with other resources.
3. They have frequently changing formats and access methods.
4. The need for complex analysis of this heterogeneous data, beyond simply bringing it all together with links, which can be complicated enough in itself.

Information Retrieval Theory Overview

According to ISO 2382/1 (1984) information retrieval (IR) is defined as:

...[the] actions, methods and procedures for recovering stored data to provide information on a given subject.

In practical terms, this translates to "given a document collection and a query, retrieve relevant documents matching the query". (Think of "document" as a container for information you want to find — it could be a web page, a database record, a file — not necessarily an article.)

The central problem of IR is the analysis and measurement of the relevance of the stored information, i.e. the relation between requested information and retrieved information. In other words, given an information inquiry, the IR system has to check whether the stored information is relevant to the inquiry.

Traditionally, this problem has been solved by organizing the database that is used for the search as an inverted file of the significant character strings occurring in stored texts, i.e. the inverted file specifies where the strings occur in the texts. Normally, the strings used are natural language words excluding determiners, conjunctions, prepositions and the like (which are known as stop-words). An inquiry to an IR system is then composed of these character strings combined by Boolean operators and some specific additional features such as contextual or positional operators. Since there is no linguistic analysis of the semantics of the stored texts or of the inquiries, IR systems are mostly domain independent. (Here "domain" = "subject matter" or "topic.")

We shall overview some technical notions from the science of information retrieval. While these apply primarily to text, they can be extended to data or sequences.

Recall and Precision. There are two fundamental measures of the quality of information retrieval:

- Recall: What percentage of relevant documents are retrieved. If a document collection contains 1000 documents relevant to a query, and 400 have been retrieved, the recall is 40%.
- Precision: What percentage of retrieved documents are relevant. If, when retrieving those 400 documents, 500 documents were retrieved in all, the precision is 400/500 or 80%.

Recall cannot easily be measured, however, because we normally do not know how many relevant documents are in a collection.

Keywords, Vocabulary and Thesauri. One problem with IR is that a text word (a "keyword") doesn't often convey the relevant meaning. Also, sometimes keywords match irrelevant hits. Consider the following examples:

1. Is a "transmembrane" protein the same as a "membrane" protein? Probably, yet the word "transmembrane" will not match "membrane" in a keyword system.
2. How many meanings does the word "domain" have? Can a query with that word match an irrelevant hit?

Sometimes, the vocabulary problem in #1 can be overcome with an online thesaurus. PubMed uses MeSH terms (from the Unified Medical Language System, also known as UMLS). When we look at PubMed we'll see how the query is expanded with other terms to make it more comprehensive and to make it match more relevant documents using alternate query terms. However, a thesaurus (with query expansion) can create problems by retrieving too much information. (A thesaurus would be used to add additional words for related senses to a query = query expansion.) The use of a thesaurus could increase recall, but why can it also degrade precision? Hint: see question #2 above.

Sensitivity and Specificity. Another way to examine the accuracy of retrieval results is by looking at the sensitivity and specificity. TP = true positives (relevant and returned by appropriate query). FP = false positives (not relevant but returned by the query). TN = true negatives (not relevant and not returned). FN = false negatives (relevant but not returned by the query when they should have been returned). (Sensitivity and specificity can be used to measure the accuracy of prediction programs of any type.)

- Sensitivity = TP / (TP + FN)
- Specificity = TP / (TP + FP)

Exercise

Let's say there are 100 records that are relevant to a query. A user retrieves 50 of them and decides that 5 are relevant. What is the precision? What is the recall?

If your system obtains data records and classifies 50 of them as positive and 30 as negative, and you determine that of the 50, 10 were false positives, and of the 30, 5 were false negatives. What is the sensitivity? What is the specificity?

Nucleic Acid Research Annual Database Issue

One important journal issue you should be aware of is the annual Database issue for NAR. You can find 2020 here https://academic.oup.com/nar/issue/48/D1. Information in this course has been drawn from a number of articles that have appeared in the database issue over the years, and they can be a source of inspiration and/or related work for your project.

Overview of NCBI Resources

A lot of your time using biological databases will be spent on obtaining data from them. This may be text retrieval, SQL, sequence alignment/searching, browsing, or writing your own programs to access data. NCBI provides many useful resources, some of which we look at now.

NCBI

The National Center for Biotechnology Information is a major Internet resource for databases and tools (https://www.ncbi.nlm.nih.gov/). This week we'll cover Entrez, sequence databases, Gene, RefSeq, Taxonomy, and PubMed. Due to the rapidly changing nature of bioinformatics resources, information found in textbooks is frequently out of date. Consequently, for more recent information use the Help links at NCBI for the various resources as necessary to learn how to search and use these databases (although sometimes the help and tutorials are not current either, as you may see). Links to databases are shown under "Popular Resources" on the NCBI homepage. Links can also be found in the Search "All Databases" dropdown list. I also have found the support staff at NCBI to be responsive to emails.

NCBI has many, many resources and it can be difficult to keep track of them all and we will not cover them all. But, for those that we do visit (and any others you are interested in), approach them from the point of view of a **database designer**. Look at how information is accessed, presented, and interconnected, and think about how these factors contribute to the user's ability to answer biological questions.

Entrez

Entrez is the main method for accessing NCBI resources. It is not a database itself but a way of searching the collection of databases at NCBI.

Exercise

Visit the main page at NCBI (https://www.ncbi.nlm.nih.gov/) and enter a search for dscr1. (That last char is the number "1".) You'll see the number of matches ("hits") in all the various databases. Take a few moments to click on some of these just to see what they give you. N**otice how the databases are integrated with each other**. This interlinking is essential for deriving full advantage of the information at NCBI (and other sources). When you work on your own project this semester, consider how your data elements relate to each other and other data.

GenBank

Nucleotide sequences are generated from a real molecule in a lab somewhere. Because they are determined experimentally, the record associated with them will describe information about how and why the sequence was created. These sequences are deposited into one of the major repositories: GenBank, DDBJ, or EMBL. These three organizations share their sequences with each other daily. The coding sequence (if any) for the nucleotide sequence is used to generate the associated protein sequence.

Genbank (https://www.ncbi.nlm.nih.gov/genbank/) can be searched via Entrez by selecting Nucleotide as the database. Some information about Genbank: it organizes its information in 18 divisions (12 based on taxonomy and 6 for high-throughput-generated sequences). Accession.version numbers are assigned by Genbank staff when the sequence is received. Sequence data includes expressed sequence tags (ESTs), genome survey sequences (GSSs), and whole-genome shotgun (WGS) sequencing sequences. Sequence data in Genbank has biological annotations and bibliography. Genbank data can be obtained via Entrez, BLAST, via links from other resources in NCBI and out, and downloaded via FTP.

Genbank has existed for a long time, but has not remained static. One big change that occurred a few years ago was the phasing out of a numbering system (GI numbers) that had been used at NCBI since 1994 to refer to a specific version of a sequence record. Versioning of accession numbers was added in 1997, and this numbering scheme of accession.version was retained and is the preferred identifier for sequence records. This removal of a commonly-used identifier was a major change, and any system that relied on GI numbers had to be modified to use accession.version instead (or some other scheme of the system's devising). This highlights the issue of systems being dependent on how databases organize their information and potential problems that could arise if there is a database change.

There was another change (that occurred a few years before the GI number change) that highlights another issue. The decision was made regarding indexing and assigning identifiers to bacterial strains – or, more importantly, when not to do so. In the case of protein sequences, a “WP”-prefix identifier was used for identical protein sequences. It is important to note that a WP-protein identifier could be associated with more than one nucleotide coding sequence. This situation relates to the issue of determining what a unique sequence is and what a unique record should be, as well as how it relates to other records in other resources. When you work with a new resource, one thing to determine is what the database considers to be a unit or record of information and what the resource does with uniqueness/redundancy. This topic is continued under the RefSeq section below.

Data Model

We look now at what is contained in a GenBank record in the GenBank flat file format, which is what you will see when viewing GenBank information through the NCBI website (it is a "display format"). Please **click** on this link: https://www.ncbi.nlm.nih.gov/genbank/samplerecord/ to view an example. Some important entities in this model are:

- LOCUS – this includes several sub-elements: the locus name (what is given to the location for this sequence), sequence length, molecule type, GenBank division, and modification date.
- ACCESSION
- VERSION
- KEYWORDS
- SOURCE – a free text field describing the organism and possibly other information.
- ORGANISM – formal scientific name of the organism.
- REFERENCE – Publication(s) discussing the data in the GenBank Record
- FEATURES – this has many sub-elements. CDS, for example, indicates a coding sequence and will be associated with a protein identifier linked to the Protein database. The definition of these features is described at http://www.insdc.org/documents/feature_table.html.
- ORIGIN – normally this contains the text of the sequence

GenBank uses the ASN.1 format internally (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808980/). As explained at https://www.ncbi.nlm.nih.gov/Structure/asn1.html, "ASN.1, or Abstract Syntax Notation One, is an International Standards Organization (ISO) data representation format used to achieve interoperability between platforms. NCBI uses ASN.1 for the storage and retrieval of data such as nucleotide and protein sequences, structures, genomes, PubMed records, and more. It permits computers and software systems of all types to reliably exchange both the data structure and content." (See example under Gene, below.)

On a different note (although related because it deals with the format of information), if you view a live GenBank page such as this one, https://www.ncbi.nlm.nih.gov/nucleotide/U49845, and click on the FASTA link near the top, you can obtain the sequence data in FASTA format. The **FASTA format** ("fast A") is a simple way to represent a sequence. The first line starts with a > and has some descriptive information about the sequence. The remaining lines are the actual sequence. Here is the start of U49845.

```
>gi|1293613|gb|U49845.1|SCU49845 Saccharomyces cerevisiae TCP1-beta gene,
partial cds; and Axl2p (AXL2) and Rev7p (REV7) genes, complete cds
GATCCTCCATATACAACGGTATCTCCACCTCAGGTTTAGATCTCAACAACGGAACCATTGCCGACATGAG
ACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAAGCCGCTGAA
GTTCTACTAAGGGTGGATAACATCATCCGTGCAAGACCAAGAACCGCCAATAGACAACATATGTAACATA
TTTAGGATATACCTCGAAAATAATAAACCGCCACACTGTCATTATTATAATTAGAAACAGAACGCAAAAA
TTATCCACTATATAATTCAAAGACGCGAAAAAAAAAGAACAACGCGTCATAGAACTTTTGGCAATTCGCG
```

Protein (GenPept)

This database contains information from several sources, such as PDB (we'll visit this at a later time) and translations from DNA sequences' coding regions. Be aware of the difference between nucleotide and protein sequences — one set is experimentally derived and the other is normally computationally derived. The distinction between experimentally derived and computer-generated data may impact how confident a user can be in results obtained from a database.

Gene

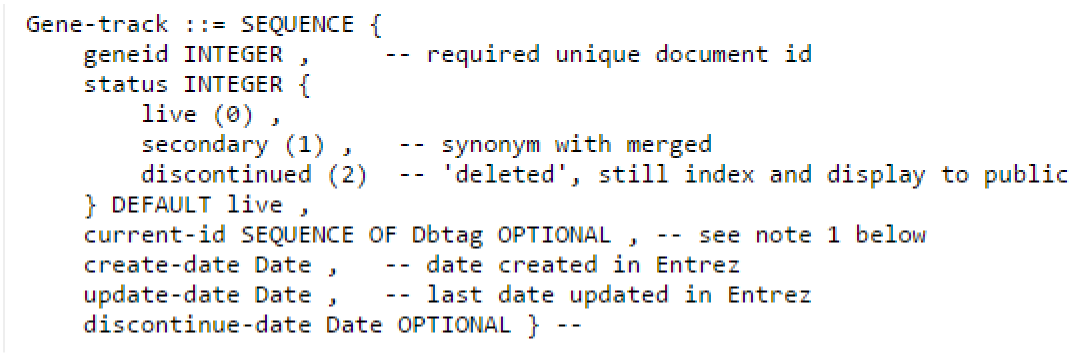
Gene (https://www.ncbi.nlm.nih.gov/gene/). Gene provides a single query interface to *curated* sequences and descriptive information about genetic loci for certain organisms (those with "reference sequences"). Often curated data is more reliable than computationally generated data.

Data Model

According to https://www.ncbi.nlm.nih.gov/books/NBK169435 :

> Gene has a simple data model. Once the concept of a gene is defined by sequence or mapped location, it is assigned a unique integer identifier or GeneID. Then data of particular types are connected to that identifier. These types include sequence accessions, names, summary descriptions, genomic locations, terms from the Gene Ontology Consortium, interactions, related phenotypes, and summaries of orthology.

The entire Gene database can be extracted as binary ASN.1, and can be converted to XML. We mentioned the ASN.1 format earlier. This page shows part of a Gene record in ASN.1 format: https://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/lxr/source/src/objects/entrezgene/entrezgene.asn. An excerpt is shown below.



```
Gene-track ::= SEQUENCE {
    geneid INTEGER ,      -- required unique document id
    status INTEGER {
        live (0) ,
        secondary (1) ,    -- synonym with merged
        discontinued (2)   -- 'deleted', still index and display to public
    } DEFAULT live ,
    current-id SEQUENCE OF Dbtag OPTIONAL , -- see note 1 below
    create-date Date ,    -- date created in Entrez
    update-date Date ,    -- last date updated in Entrez
    discontinue-date Date OPTIONAL } --
```

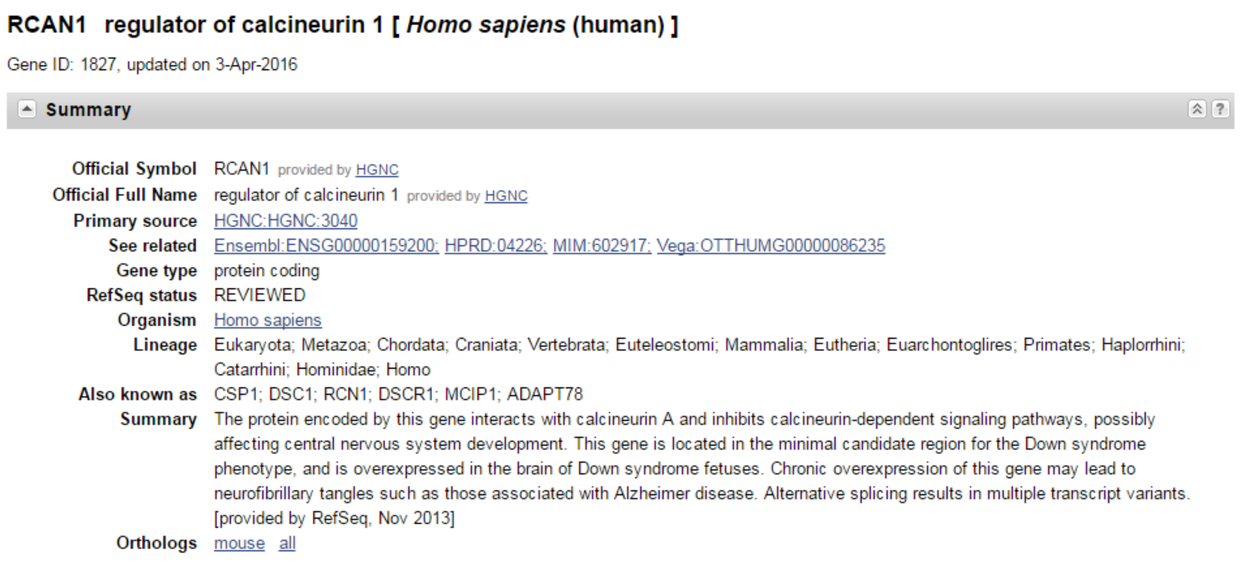
A more useful data model is what is presented to the end user when looking at a Gene record online (see the exercise below).

Exercise

Visit the URL for Gene (https://www.ncbi.nlm.nih.gov/gene/), and search on dscr1. (Note you can also click from the Entrez search results page). Use the Top Organisms on the right to limit the results to just Homo sapiens.

Notice in the results list the Name/Gene ID and the Aliases columns. This setup helps alleviate the issues with naming that crop up frequently when dealing with genes. If you create a gene resource it is imperative that you make use of the official symbol but also accommodate other frequently used names.

Click on the RCAN1 record and **view** the record. We can see that the data model includes the Gene ID (1827), the update date (3-Apr-2016 at the time this section of the Module Lecture Content was written), the Official Symbol (RCAN1), etc.



RCAN1 regulator of calcineurin 1 [*Homo sapiens* (human)]

Gene ID: 1827, updated on 3-Apr-2016

Summary

Official Symbol	RCAN1 provided by HGNC
Official Full Name	regulator of calcineurin 1 provided by HGNC
Primary source	HGNC:HGNC:3040
See related	Ensembl:ENSG00000159200; HPRD:04226; MIM:602917; Vega:OTTHUMG00000086235
Gene type	protein coding
RefSeq status	REVIEWED
Organism	Homo sapiens
Lineage	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo
Also known as	CSP1; DSC1; RCN1; DSCR1; MCIP1; ADAPT78
Summary	The protein encoded by this gene interacts with calcineurin A and inhibits calcineurin-dependent signaling pathways, possibly affecting central nervous system development. This gene is located in the minimal candidate region for the Down syndrome phenotype, and is overexpressed in the brain of Down syndrome fetuses. Chronic overexpression of this gene may lead to neurofibrillary tangles such as those associated with Alzheimer disease. Alternative splicing results in multiple transcript variants. [provided by RefSeq, Nov 2013]
Orthologs	mouse all

What other information is available further down the page? Notice you can determine where the gene occurs in the genome, information about transcripts, references, and much more. What other resources are linked to (there are many – both within the information for the gene and under Related Information to the right)?

If you wish, view http://www.youtube.com/watch?v=RHz2nZbzjpA&list=PL8FD4CC12DABD6B39 and see how you can use the Gene record as a jumping off point for finding other information, in this case the genomic sequence.

RefSeq

According to NCBI, "The Reference Sequence (RefSeq) collection aims to provide a comprehensive, integrated, non-redundant set of sequences, including genomic DNA, transcript (RNA), and protein products, for major research organisms." That is, often there are many sequence records entered for a given gene, transcript, or protein because it came from different tissues or research centers, etc. (Computer scientists take note of this redundancy, which can cause confusion until you realize it exists.) RefSeq aims to provide one sequence record for a given gene, transcript, or protein. Not all organisms have reference sequences, however.

From “Search NCBI Databases Using Entrez”, RefSeq section in “Current Protocols in Bioinformatics, Unit 1.3”:

> “RefSeq entries are distinguished from other entries in GenBank through the use of a distinct accession number series. RefSeq accession numbers follow a “2 + 6” format: a two-letter code indicating the type of

> reference sequence, followed by an underscore and a six-digit number… It is important that the reader understand the distinction between the "N" numbers and "X" numbers. The former represent actual, experimentally determined sequences, whereas the latter represent computational predictions derived from the raw DNA sequence." [Note: identifiers may have more than 6 digits now.]

One of the database issues we'll see in this class deals with deciding what kind of information is in a database -- what is a unit of information, how granular is it? What constituted redundant information and what do we do with it?

RefSeq has this issue with regard to prokaryotic sequences coming from metagenomic sequencing projects. Often many of these sequences are identical -- should they all be given a unique record and accession number? In the past, they were, however, this situation changed in 2013.

Another issue we'll look at in more detail later in the course is the problem with large amounts of data -- also something that needs to be addressed with metagenomic sequencing output.

The following is taken from https://www.ncbi.nlm.nih.gov/refseq/about/nonredundantproteins and explains how RefSeq now contains one protein record that represents all the different sequences that previously would have been individual records. So -- when you go to obtain information from a resource, be sure you know what the granularity (level of detail) and scope (what information is covered) of an individual record is. And be aware that different resources may organize similar (or the same) information differently.

> "A new type of RefSeq protein record which represents non-redundant protein sequences was introduced in mid-2013. This record type was introduced to address a growing issue with redundancy in the Prokaryotic RefSeq protein dataset that coincided with a significant increase in bacterial genome submissions from individual isolates and closely related bacterial strains. For example, a large number of high-quality bacterial genomes may be submitted during a disease outbreak. The submitted sequences may reflect pathogen evolution during the course of the outbreak but the majority of the encoded proteins from these genomes may be identical to each other. As RefSeq includes these genomes, per community requests, this resulted in increased redundancy. By representing identical proteins using a single non-redundant protein accession number (with the prefix 'WP_'), redundancy in the database is significantly reduced.
>
> "When the NCBI genome annotation pipeline annotates a bacterial protein that is 100% identical and the same length as an existing non-redundant protein, NCBI will annotate that protein on the genome by referencing the WP_ accession in the annotated CDS feature. Any annotation of protein function on the genome record, such as the protein name, will be inherited from the independently-maintained non-redundant protein record. Non-redundant protein records always represent one exact sequence that has been observed once or many times in different strains or species. These records will always have a version number of '1' because the sequence will never be changed, although the name and other descriptive annotation will be maintained and updated. Individual non-redundant protein records will be suppressed if that identical protein is no longer found on any RefSeq genome
>
> "When an important representative genome exists, such as Escherichia coli K12 substr. MG1655, the reference genome will be annotated with strain-specific reference protein accessions with the NP_ or YP_ prefix. These protein records will refer to the matching non-redundant protein record (WP_) in the sequence block. Thus, reference proteome records (NP_ or YP_) will continue to define taxonomically-oriented sets of proteins and will track sequence changes over time. If a reference protein record is revised by curation and its version number changes, it will now refer to a different non-redundant (WP_) record that is identical to the updated reference protein sequence

"DEFINITION line: the definition line includes the protein name followed by an organism name in square brackets ([ ]). The organism may be identified at the level of species to super-kingdom and does not reflect strain-specific information. This line may include a prefix 'MULTISPECIES:' if the protein record is annotated on genomes from different species. If the protein is annotated on genomes from different super-kingdoms, then each super-kingdom name is provided in separate square brackets"

Exercise

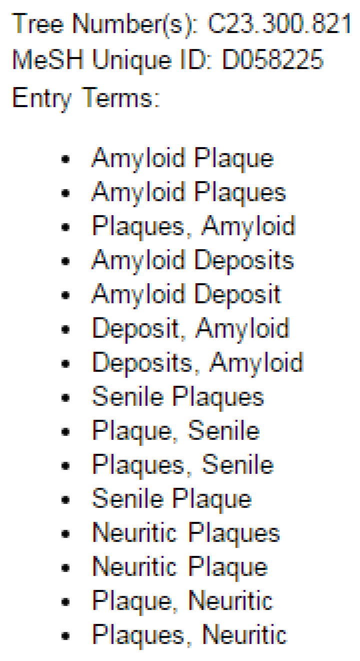
In the Gene record for RCAN1, on the right hand side under Related Information, there are links to RefSeq records. **Click** on the RefSeq Proteins section. You'll see the Genomic, mRNA, and reference proteins. Note the format of the accession numbers. Click on one or two of the links and see the kind of information available.

PubMed and PubMed Central (PMC)

PubMed is a database of citations with abstracts and links to full text, primary from MEDLINE. PMC contains full-text articles. The two resources do not cover the same material, although there is considerable overlap. Below are links to some how-to videos for searching PubMed. Please view these if you are unfamiliar with searching PubMed.

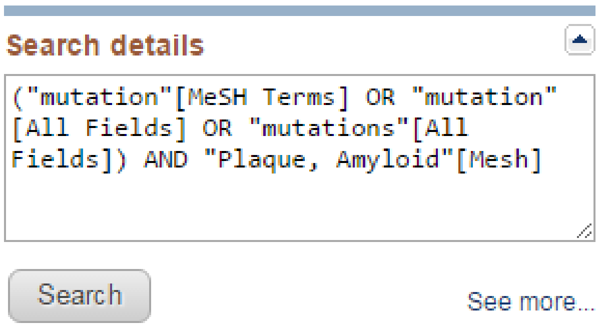
You may want to create a MyNCBI account so you can save your searches.

One important point about PubMed is that you can use MeSH (Medical Subject Headings) in your search to obtain synonyms. Below you can see a set of entry terms for one of the MeSH concepts:


Tree Number(s): C23.300.821
MeSH Unique ID: D058225
Entry Terms:

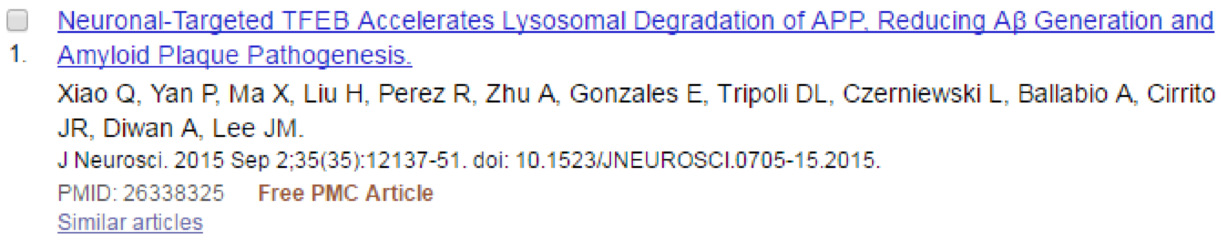
- Amyloid Plaque
- Amyloid Plaques
- Plaques, Amyloid
- Amyloid Deposits
- Amyloid Deposit
- Deposit, Amyloid
- Deposits, Amyloid
- Senile Plaques
- Plaque, Senile
- Plaques, Senile
- Senile Plaque
- Neuritic Plaques
- Neuritic Plaque
- Plaque, Neuritic
- Plaques, Neuritic

The following shows the search details for a PubMed query using the MeSH concept:



```
("mutation"[MeSH Terms] OR "mutation"
[All Fields] OR "mutations"[All
Fields]) AND "Plaque, Amyloid"[Mesh]
```

And, finally an article that was found with the query in which “Amyloid Plaque” was in the title (but not explicitly in the search):



1. Neuronal-Targeted TFEB Accelerates Lysosomal Degradation of APP, Reducing Aβ Generation and Amyloid Plaque Pathogenesis.
Xiao Q, Yan P, Ma X, Liu H, Perez R, Zhu A, Gonzales E, Tripoli DL, Czerniewski L, Ballabio A, Cirrito JR, Diwan A, Lee JM.
J Neurosci. 2015 Sep 2;35(35):12137-51. doi: 10.1523/JNEUROSCI.0705-15.2015.
PMID: 26338325 Free PMC Article
Similar articles

Exercise

The following videos give useful information to improve your searching of PubMed.

http://www.youtube.com/watch?v=uyF8uQY9wys&list=PLBD13A2628C7A9965&index=1 for information about how to use MESH in searching.

http://www.youtube.com/watch?v=dncRQ1cobdc&list=PLBD13A2628C7A9965&index=2 for information on performing an advanced search. Notice that field searches result in the name or tag of the field in [] being added to the search term. Also note the use of saved searches in building new searches.

http://www.youtube.com/watch?v=696R9GbOyvA&list=PLBD13A2628C7A9965&index=4 and get comfortable using filters. Also note the use of the Search Details box and what information is shown there. Do some practice queries of your own.

Taxonomy

Taxonomy (https://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/). One method for accessing data is by querying — another main way is by browsing. Taxonomy data is particularly appropriate for accessing via browsing. But note that you can also access it with a query just as you can other NCBI databases. In addition, the site has direct links to model organisms at the top of the landing page. Often providing multiple methods of obtaining data from a resource is important — this allows researchers to use intuitive approaches to retrieving the information they want, rather than spend time figuring out how to frame their question to match the input requirements of the database.

One database issue that the Taxonomy database has to address is the names of things, which is surprisingly complex. The following is from the article, “The NCBI Taxonomy database”, http://nar.oxfordjournals.org/content/40/D1/D136.full. The Taxonomy DB started in 1991. The taxonomy is based on current scientific understanding and is intended to be a "phylogenetic taxonomy" that "corresponds with the evolutionary history of the tree of life." It makes use of some general taxonomy database projects as well as specialized taxonomies for particular branches (such as Algaebase for algae). You can use the taxonomy identifiers for organisms in Entrez searches to restrict your results.

Initially NCBI (GenBank), EMBL, and DDBJ (the three major sequence databases) had their own taxonomy nomenclature and classifications. The 3 databases shared sequences but "the source organism nomenclature and taxonomic classifications were inconsistent and were updated irregularly." Because Entrez links databases together it needed a consistent taxonomy across all sources. This quote is too good to miss:

> "The first step was to shuffle together the taxonomies from each of the contributing databases, each of which covered a somewhat different set of species with often very different internal classifications. The end result of this process was a hideous abomination..."

After this the 3 database members resolved these differences, namely by taking care of synonyms, misspellings, and alternate classifications. Other issues that needed to be handled are:

- Duplicate names -- common. A black darter is a fish and a dragon fly.
- Duplicate names -- scientific. Although scientific names are a type of controlled vocabulary, there is no guarantee that there are no duplicates. One example given is that *Lestoidea* is a damselfly genus and a superfamily. Another is *Gnathostomata* which is a superclass (jawed vertebrates) and a superorder (sand dollars). Yet another is Bacillus -- there are bacteria of this genus and this term also applies to some types of insects. Due to the duplication of a large number of genus names, there are name conflicts with some common species names (such as *americana*). There are 6 binomial duplications; consequently, the authority of the name is also used to resolve the conflict.
- Two names -- fungi. Fungi can have two different (and correct) scientific names, one based on anamorph names and the other on teleomorph names.

Also, GenBank no longer gives taxonomy IDs for strains of microbes, with some exceptions. Instead, the taxonomy id will be at the species level for new strains.

What should we take away from all this?

- Make sure you know what the entities in a database mean and how they were derived.
- Understand how they are named.
- If you try to integrate data, you need to understand if names are duplicated (particularly between different sources).
- If there are duplicates, do they have the same meaning (semantics) or not?
- You may be expecting that everything has one unique name -- but it may not.
- Identifiers may not be consistently given for all data in a database – some records may have identifiers at one level (strain), others at a different level (species only).

Additional Databases

Although an entire course could be focused on NCBI, we are going to look at a variety of databases and resources from other providers as we progress through the class. We’ll look at two more this week.

GeneOnology

GO http://www.geneontology.org/. The goal of the Gene Ontology Consortium is to produce a dynamic controlled vocabulary to describe gene products with regard to three sub-ontologies: “Molecular Function” (molecular-level activities performed by gene products), “Biological Process” (the larger processes that occur due to multiple molecular functions), and “Cellular Component” (location relative to structures of a cell). GO annotations are supported by evidence (there are many different evidence codes). GO terms are used in **many** other databases to annotate information, and this brings uniformity of description (semantics and syntax) to any databases that use its vocabulary. GO has grown as knowledge of gene and protein roles in cells is accumulating and changing. In this course, you will see a number of resources that make use of GO terms. There are also tools (which we won’t be covering) such as those that do enrichment analysis as part of analyzing a gene set.

In the NAR article, “Expansion of the Gene Ontology knowledgebase and resources” https://academic.oup.com/nar/article/45/D1/D331/2605810/Expansion-of-the-Gene-Ontology-knowledgebase-and, the authors point out that due to the large size of biomedical datasets, it is necessary to rely on knowledge -- such as functional annotation of gene products -- that is stored in a manner that can be fed into algorithms. Using an ontology like GO enables the storage of this "computable knowledge".

One form this computable knowledge takes is as a Gene Ontology annotation. A GO annotation consists of the identification of a gene or gene product and its association with a particular GO term (identifier), and the evidence that supports making this association. It is very important that the gene or gene product name/identifier in the annotation refer to a single, common identifier in a database. This avoids confusion and ambiguity regarding which gene/gene product is being annotated. In addition it is equally important that evidence for the annotation is provided. The evidence is frequently a specific publication -- when it is, the annotation includes the PubMed identifier. The evidence is also given a code that describes the type of evidence.

There are 3 general categories of evidence: manually asserted with direct evidence; manually asserted with indirect evidence; and automatically asserted. Here is an example of a manual evidence code with direct evidence from the 2017 article:

> "For example, an association between a gene product and its subcellular localization as determined by immunofluorescence would be supported by the Inferred from Direct Assay (IDA) evidence code, a subtype of EXP evidence. Annotations with direct experimental evidence are created by biocurators, PhD-level experts trained in computational knowledge representation, who read peer-reviewed literature and create GO annotations as justified by the evidence presented in those articles."

Some annotations are created because they have been transferred due to sequence similarity. Sometimes these transfers are manually performed. Here is an example from the article:

> "[GO has a tool in which] a biocurator can view all experimental annotations for genes in a gene family, and use this information to infer annotations for uncharacterized members of the family. The biocurator creates an explicit model of gain and loss of gene function at specific branches in a phylogenetic tree of the family. This model is used to infer new annotations (i.e. not overlapping with experimental annotations) for genes in the family. Phylogenetically based annotations are denoted by the IBA (Inferred from Biological Ancestry) evidence codes. Each inferred annotation can be traced to the direct experimental annotations that were used as the basis for that assertion."

At other times the annotations are transferred through a strictly automated process without manual oversight based on sequence similarity, the mapping of Enzyme Commission identifiers to GO identifiers, or other automated methods. These annotations are given the code IEA. Thus, the evidence codes indicate whether the annotation was asserted manually or in an automated fashion with or without manual review.

As discussed in this NAR article **"**Gene Ontology Annotations and Resources" http://nar.oxfordjournals.org/content/43/D1/D1049.full, GO requires submitted information for an annotation have identifiers for the gene products that can be mapped to UniProtKB or NCBI identifiers (essential to make sure the annotation is referring to the same thing in those instances when it is referring to the same thing). A recent area of focus for GO deals with multi-organism processes, which required adding the ability to track relationships (such as symbiont_of, host_of, parasite_of, and vector_for) between the organisms involved in the multi-organism process to GO.

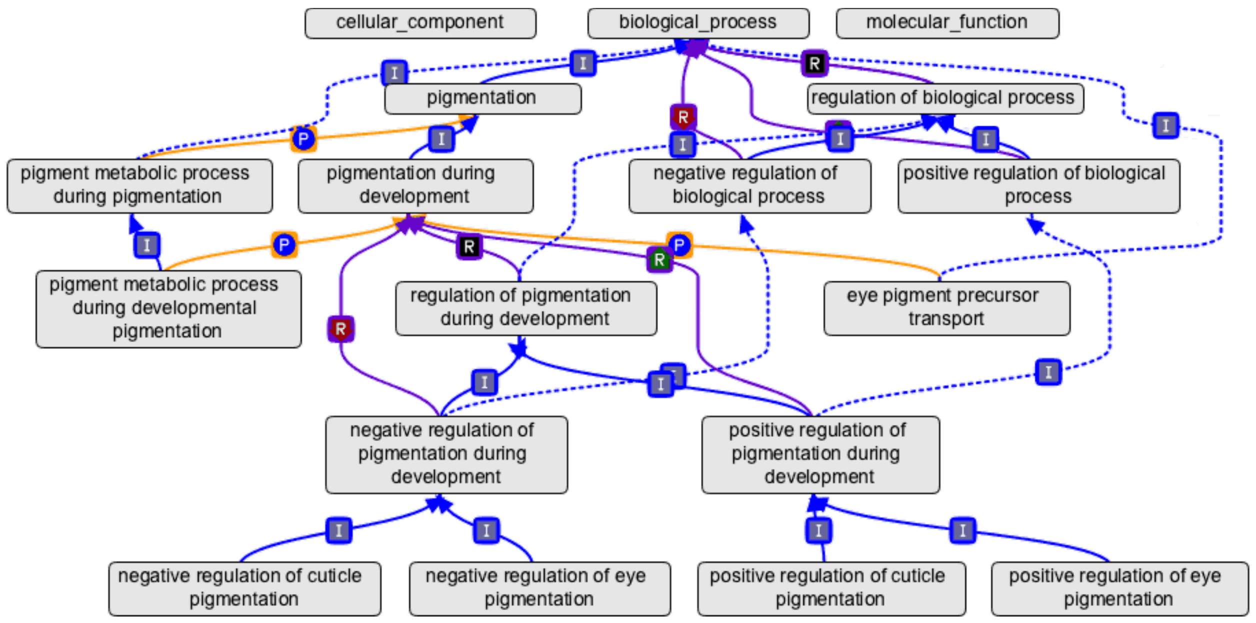
This article points out that shared vocabularies are needed for unifying data across databases, but knowledge changes over time, and different people may view information differently. Manual review of terms, improved tools, extension of the GO annotation capability, and use of subject-area experts helps make the resource up-to-date, accurate and consistent. (If you are interested in semantic web technologies, note too that GO has an OWL version.)

In the past, GO used a relational database (MySQL). Currently, GO uses the OBO file format for the ontology (an example is given below). Files in this format are converted to XML for users who would like the data this way. In addition, data at GO can be obtained in RDF (which is written in XML) and OWL (also uses XML).

Thus, this database provides an example where we have multiple different ways of representing the same information.

Data Model

GO contains terms organized as a graph (terms are nodes). The following diagram is from an older page at GO, and shows a small part of the ontology for biological process. Here, just be aware that the terms are linked together, typically (although not always) hierarchically, and some terms can have more than one parent.


cellular_component
biological_process
molecular_function
pigmentation
regulation of biological process
pigment metabolic process during pigmentation
pigmentation during development
negative regulation of biological process
positive regulation of biological process
pigment metabolic process during developmental pigmentation
regulation of pigmentation during development
eye pigment precursor transport
negative regulation of pigmentation during development
positive regulation of pigmentation during development
negative regulation of cuticle pigmentation
negative regulation of eye pigmentation
positive regulation of cuticle pigmentation
positive regulation of eye pigmentation
I
P
R

The following is an example of a GO term taken from the OBO formatted file:

```
id: GO:0016049
name: cell growth
namespace: biological_process
def: "The process in which a cell irreversibly increases in size over time by accretion and biosynthetic production of matter similar to that already present." [GOC:ai]
subset: goslim_generic
subset: goslim_plant
subset: gosubset_prok
synonym: "cell expansion" RELATED []
synonym: "cellular growth" EXACT []
synonym: "growth of cell" EXACT []
is_a: GO:0009987 ! cellular process
is_a: GO:0040007 ! growth
relationship: part_of GO:0008361 ! regulation of cell size
```

“Subset” indicates that a GO Term is part of a designated subset of terms. Notice the “is_a” and “relationship: part_of” attributes. These indicate links of specific types between GO terms in the graph. The above entry does not contain an example of database cross reference, which is used to connect to an external definition, such as to an Enzyme Commission EC number. In addition, there is a way to indicate that a GO term is obsolete.

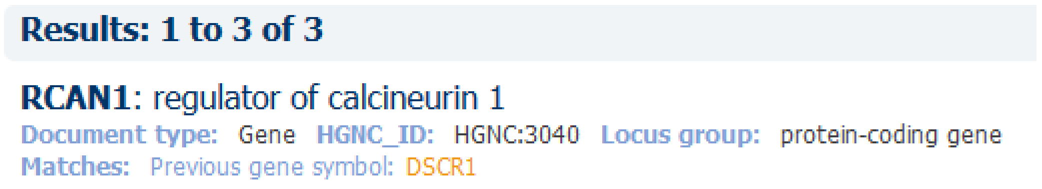
In the old relational model for GO, the links between nodes were handled in the tables term, term2term and graph_path. The graph_path table contained precomputed paths for all terms to their ancestors (the transitive closure). It was used for efficiency in MySQL to make it possible to retrieve something like “all DNA binding genes” more quickly than traversing through the term2term table. I bring this up to point out that graphs can be represented in relational tables.

The object-oriented version in OBO and the relational version are not identical, because there is no need for a graph_path in the OBO version. I would say in this situation that the logical data model is the same, but the physical data model and the format of the stored data differ.

HUGO

The HUGO Gene Nomenclature Committee has a website http://www.genenames.org/ that you can use to determine the approved name for a gene. Because of the importance of using the same name to refer to the same thing, be aware of this site for human gene names.

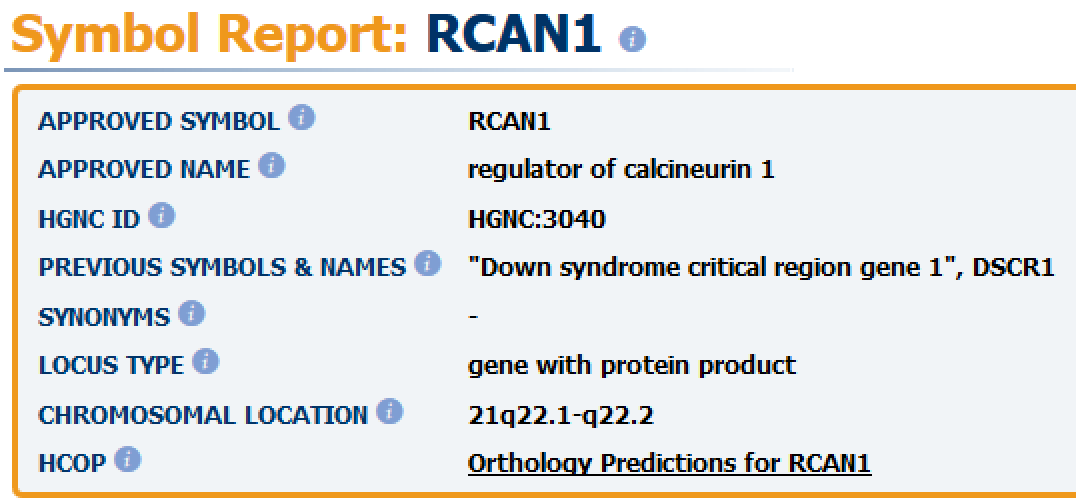
You can search HUGO, for example, for DSCR1, and it will return the approved symbol:



Results: 1 to 3 of 3

RCAN1: regulator of calcineurin 1
Document type: Gene HGNC_ID: HGNC:3040 Locus group: protein-coding gene
Matches: Previous gene symbol: DSCR1

The following figure provides a part of the RCAN1 record.



Symbol Report: RCAN1

APPROVED SYMBOL	RCAN1
APPROVED NAME	regulator of calcineurin 1
HGNC ID	HGNC:3040
PREVIOUS SYMBOLS & NAMES	"Down syndrome critical region gene 1", DSCR1
SYNONYMS	-
LOCUS TYPE	gene with protein product
CHROMOSOMAL LOCATION	21q22.1-q22.2
HCOP	Orthology Predictions for RCAN1

Wrap Up

We covered a lot of ground this week focused around a few themes, including:

- What is a unique record? What data should be given an identifier?
- Data Models
- Data integration
    - Handling issues with names: consistency, duplications, multiples, and semantics
    - Linking between resources (“loose” integration)
- Accessing information
    - Querying a collection of databases – Entrez
    - Text search (IR)
    - Browsing