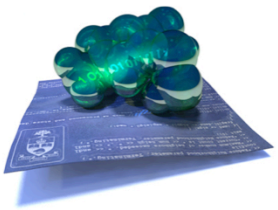
A
Bioinformatics
Course

Storing Data



Boris Steipe

department of biochemistry – department of molecular genetics
University of Toronto

Storing Data as Bits and Bytes

All data is stored as patterns of 0 and 1 – and interpreted as binary numbers.

The integer 123456 is stored in 32 bits (4 bytes) as:
[00000000][00000001][11100010][01000000]

As a float 1.23456e05 is stored in the same way

As text it is stored in 7 bytes per character
[1][2][3][4][5][6]
[011 0001][011 0010][011 0011][011 0100][011 0101][011 0110]

In the computer, all data is represented as bits.

It depends on the context whether the bits are interpreted as an element of **data** - integer, floating point number or text, or even a complex data structure such as an image or a formatted word-processor document, or an **instruction** to the processor.

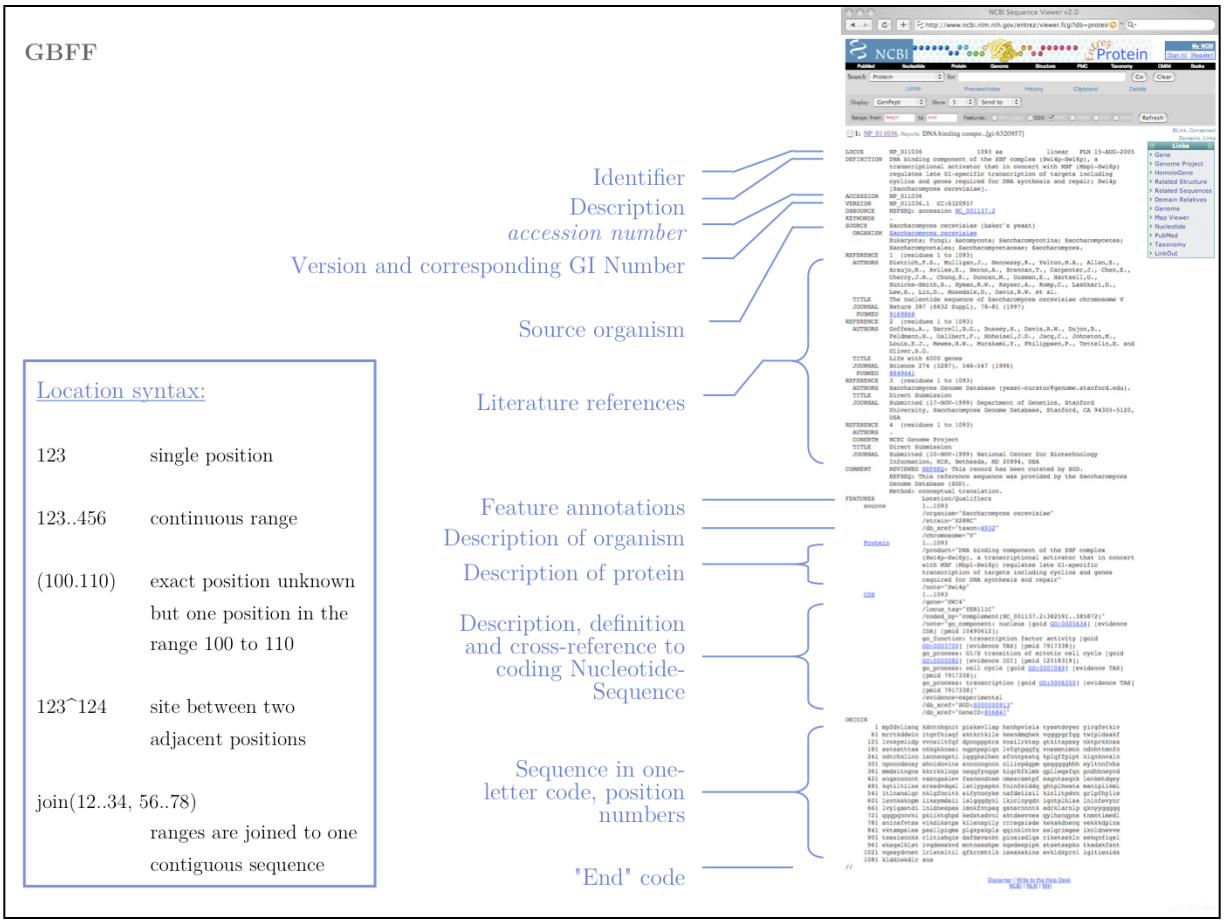
REPRESENTATION

Sequences are represented as “strings”.
Simplest, most common representation: FASTA format

In order to store complex data, we need to agree on a **data format**.

We can roughly divide our options into “flat file formats” and structured data grammars. In a flat file format, we have ad-hoc rules that identify the semantics of individual lines. One such example is the FASTA format, named for a legacy sequence alignment program of the 1980s.

The FASTA format is a simple, readable representation but it is not designed for extensive annotations. Because of its simplicity, it is still the *lingua franca* of bioinformatics file formats; practically bioinformatics tools and Web services that operate with sequences are able to read and write FASTA formatted sequences.



Genbank has its own GenBank Flat file Format (GBFF).

The amount of annotation can be extensive, providing cross-references, lists of features, sequence translations etc. You should be familiar with the syntax of location identifiers and you should be familiar with the standard contents of a Genbank record.

Parsing

In order to use data records that have more than one field, the records have to be "parsed".

- FASTA has *ad-hoc* rules: lines starting with ">" are headers, the following lines contain the data ... however, these rules are not stored with the data itself.
- Other biological data types are based on more complex (but still *ad-hoc*) file formats: e.g. GenBank Flat Files, PDB files, BLAST output files ...
- Documentation, maintenance and correct parsing are a problem. File formats are frequently changed, breaking all software tools that assume a particular format.

One would wish for precise format specifications published with every data record, one would wish for self-describing formats, one would wish for ontologies, and controlled vocabularies and one would wish for **validated parser code in common languages to be available with any database download** ... in reality the support of database users is generally quite poor. Why? This probably results from the fact that one can't publish such support – or quality assurance efforts in general – and one can't strengthen one's grant proposals with user support and what doesn't get funded and doesn't get done.

The economical and intellectual damage resulting from this is vast. Several lifetimes of graduate student man- and womanpower have been wasted through the need to periodically update BLAST output file parsers. And it is said that there is not a single parser that can correctly read all information in all PDB files.

R packages have made this situation a little better, but much still needs to be done.

Data Grammars solve many of the parsing problems of "flat files".

A data grammar is a language for storing data.
Parsing is rule based – not *ad-hoc*
Parsing code (and writing code) can be automatically generated (error-free!).

Examples:

ASN.1 (used since 1991 by NCBI, others)
Abstract Syntax Notation 1

XML (Widely adopted Internet standard)
Extended Markup Language

Just like in human language, rigorous syntax rules enforce that you can't use bad grammar and get away with it.

NCBI TinySeq XML example – a data grammar

```
<?xml version="1.0"?>
<!DOCTYPE TSeq PUBLIC "-//NCBI//NCBI TSeq/EN" "http://www.ncbi.nlm.nih.gov/dtd/
NCBI_TSeq.dtd">
<TSeq>
  <TSeq_seqtype value="protein"/>
  <TSeq_gi>6320147</TSeq_gi>
  <TSeq_accver>NP_010227.1</TSeq_accver>
  <TSeq_taxid>4932</TSeq_taxid>
  <TSeq_orgname>Saccharomyces cerevisiae</TSeq_orgname>
  <TSeq_defline>Mbp1p [Saccharomyces cerevisiae]</TSeq_defline>
  <TSeq_length>833</TSeq_length>
  <TSeq_sequence>MSNQIYSARYSGVDVYEFIHSTGSIMKRKKDDWVNATHILKAANFAKAKRTRILEKEVLKET
HEKVQGGFGKYQGTWVPLNIAKQLAEKFSVYDQLKPLFDFTQTDGSASPPPAPKHHHASKVDRKKAIRSASTSAIMETK
RNNKKAEENQFQSSKILGNPTAAPRKRGRPVGSTRGSRRKLGVNLQRSQSDMGFPRPAIPNSSISTTQLPSIRSTMGPQ
SPTLGILEEERHDSRQQQPQQNNSAQFKEIDLEDGLSSDVEPSQQLQQVFNQNTGFVPQQQSSLIQTQQTESMATSVSS
SPSLPTSPGDFADSNPFEERFPGGGTSPIISMIPRYPVTSRPQTSDINDKVNKYLSKLVDYFISNEMKSNKSLPQVLLH
PPPHSAPYIDAPIDPELHTAFHWACSMGNLPIAEALYEAGTSIRSTNSQGQTPLMRSSLFHNSYTRRTFPRIFQLLHET
VFDIDSQSQTVIHHIVKRKSTTPSAVYYLDVVLSKIKDFSPQYRIELLLNTQDKNGDTALHIASKNGDVVFFNTLVKMG
ALTTISNKEGLTANEIMNQQYEQMMIQNGTNQHVNSSNTDLNIHVNTNNIETKNDVNSMVIMSPVSPSDYITYPSQIAT
NISRNIPNVVNSMKQMASIYNDLHEQHDNEIKSLQKTLKSISKTKIQVSLKTLEVLKESSKDENGEAQTNDDFEILSRL
QEQNTKKLRKRLIRYKRLIKQKLEYRQTVLLNKLIEDETQATTNNTVEKDNNTLERLELAQELTMLQLQRKNKLSSLVK
KFEDNAKIHKYRRIIREGTEMNIEEVDSSLDVILQTLIANNNKNKGAEQIITISNANSHA</TSeq_sequence>
</TSeq>
```

The essence of XML (Xtended Markup Language) is the “markup of data-elements with opening and closing tags, or standalone tags, (formatted blue above); metadata can be contained within tags as attributes (formatted purple above). As well, the definition of an XML format **must** available at the URL that leads to the .dtd file (“document type definition”). You can check the TinySeq DTD at the URL:

`https://www.ncbi.nlm.nih.gov/dtd/NCBI_TSeq.dtd`

This principle makes XML a **self-describing data format**.

XML formatted files are human readable – but only ''in principle''. The abundance of tags can make navigating the data challenging in practice. The loss of readability is a trade-off for the gain in rigour. For R, XML formatted data can be parsed with the older xml package and the newer xml2 package. Use these packages and definitley resist the urge to parse XML “by hand”, using regular expressions.

Regular expressions can’t guarantee to be able to parse XML!

GenBank XML File

```
                    <Title_E_issn>0021-9258</Title_E_issn>
                  </Title_E>
                  <Title_E>
                    <Title_E_jta>HIV</Title_E_jta>
                  </Title_E>
                </Title>
              </Cit-jour_title>
              <Cit-jour_imp>
                <Imprint>
                  <Imprint_date>
                    <Date>
                      <Date_std>
                        <Date-std>
                          <Date-std_year>1989</Date-std_year>
                          <Date-std_month>3</Date-std_month>
                          <Date-std_day>15</Date-std_day>
                        </Date-std>
                      </Date_std>
                    </Date>
                  </Imprint_date>
                  <Imprint_volume>264</Imprint_volume>
                  <Imprint_issue>8</Imprint_issue>
                  <Imprint_pages>4304-4311</Imprint_pages>
                </Imprint>
              </Cit-jour_imp>
            </Cit-jour>
          </Cit-art_from_journal>
        </Cit-art_from>
      </Cit-art>
    </Pub_article>
  </Pub>
 </Pub-equiv>
</Pubdesc_pub>
<Pubdesc_comment>SEQUENCE OF 1-32.</Pubdesc_comment>
</Pubdesc>
</Seqdesc_pub>
</Seqdesc>
</Seq-descr>
</Bioseq_descr>
<Bioseq_inst>
<Seq-inst>
<Seq-inst_repr value="raw"/>
<Seq-inst_mol value="aa"/>
<Seq-inst_length>330</Seq-inst_length>
<Seq-inst_seq-data>
<Seq-data>
<Seq-data_ncbieaa>
<NCBIeaa>MKQTIFSGIQPSGSVTLGNYIGAMKQFVELQHDYNSYFCIVDQHAITVPQDRLELRKNIRNLAALYLAVGLDPEKATLFIQSEVPAHAQAGWMMQCVAYIGELERMT
</Seq-data_ncbieaa>
</Seq-data>
</Seq-inst_seq-data>
</Seq-inst>
</Bioseq_inst>
<Bioseq_annot>
<Seq-annot>
<Seq-annot_data>
<Seq-annot_data_ftable>
<Seq-feat>
<Seq-feat_data>
<SeqFeatData>
<SeqFeatData_region>Similarity</SeqFeatData_region>
</SeqFeatData>
</Seq-feat_data>
<Seq-feat_comment>"HIGH" REGION.</Seq-feat_comment>
<Seq-feat_location>
<Seq-loc>
<Seq-loc_int>
```

This small section of a Genbank file formatted as XML may illustrate what we mean when we say XML is human readable only in principle...

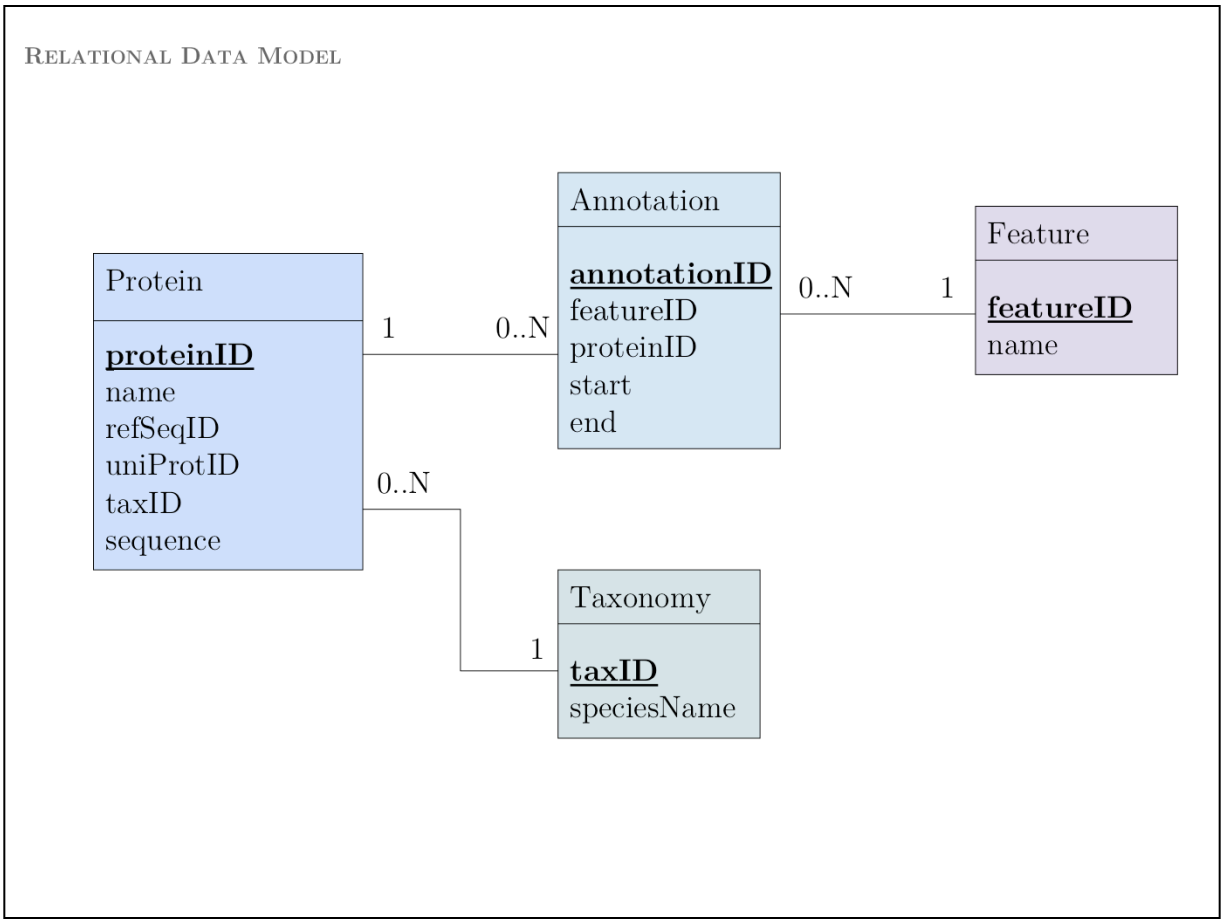
JSON

JSON is a popular data grammar for formatting key:value pairs in plain or hierarchical data structures.

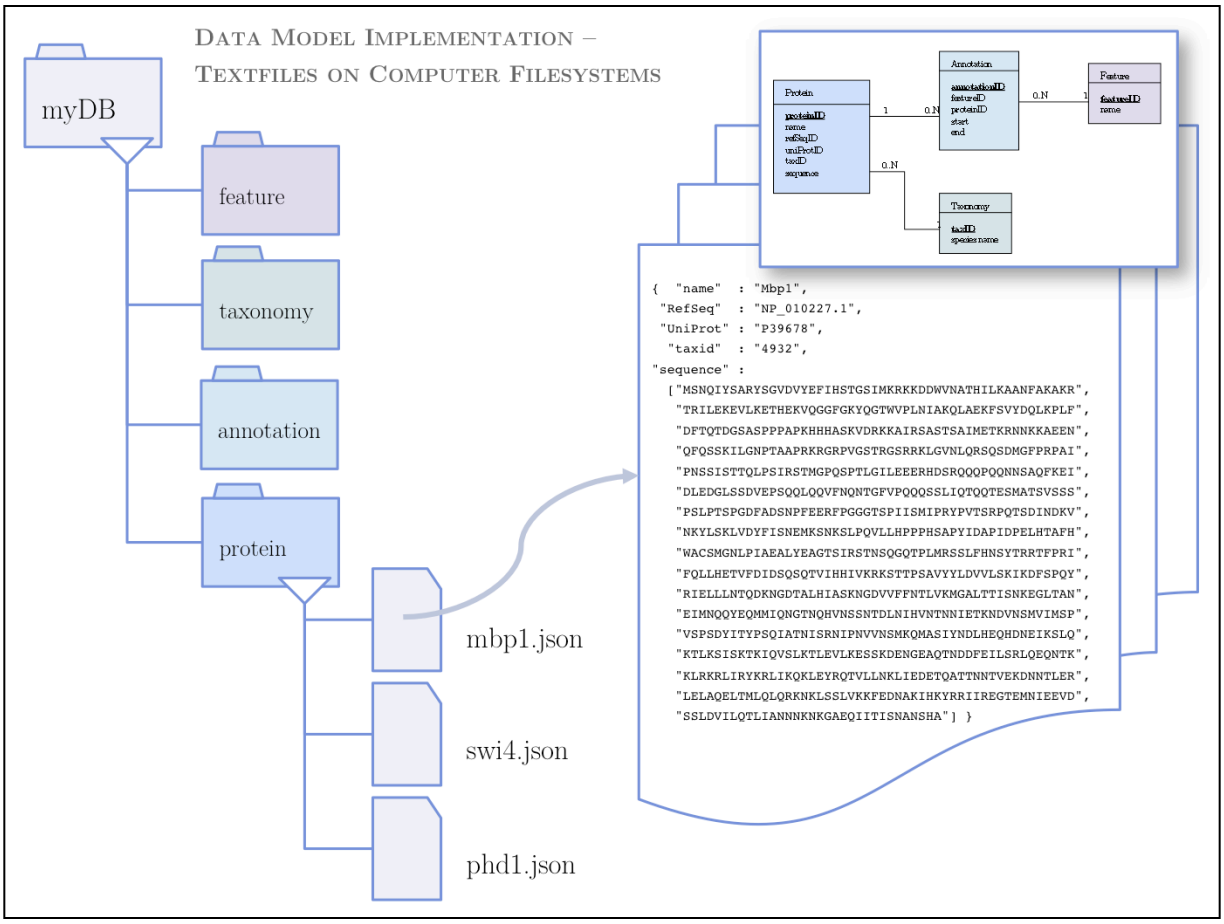
```
{  "type"  : "protein",
      "gi" : 6320147,
 "refseq"  : "NP_010227.1",
  "taxid"  : 4932,
 "orgname" : "Saccharomyces cerevisiae",
 "defline" : "Mbp1p [Saccharomyces cerevisiae]",
  "length" : 833,
"sequence" :
  ["MSNQIYSARYSGVDVYEFIHSTGSIMKRKKDDWVNATHILKAANFAKAKRTRILEKEVLKETHEKVQGGF",
   "GKYQGTWVPLNIAKQLAEKFSVYDQLKPLFDFTQTDGSASPPPAPKHHHASKVDRKKAIRSASTSAIMET",
   "KRNNKKAEENQFQSSKILGNPTAAPRKRGRPVGSTRGSRRKLGVNLQRSQSDMGFPRPAIPNSSISTTQL",
   "PSIRSTMGPQSPTLGILEEERHDSRQQQPQQNNSAQFKEIDLEDGLSSDVEPSQQLQQVFNQNTGFVPQQ",
   "QSSLIQTQQTESMATSVSSSPSLPTSPGDFADSNPFEERFPGGGTSPIISMIPRYPVTSRPQTSDINDKV",
   "NKYLSKLVDYFISNEMKSNKSLPQVLLHPPPHSAPYIDAPIDPELHTAFHWACSMGNLPIAEALYEAGTS",
   "IRSTNSQGQTPLMRSSLFHNSYTRRTFPRIFQLLHETVFDIDSQSQTVIHHIVKRKSTTPSAVYYLDVVL",
   "SKIKDFSPQYRIELLLNTQDKNGDTALHIASKNGDVVFFNTLVKMGALTTISNKEGLTANEIMNQQYEQM",
   "MIQNGTNQHVNSSNTDLNIHVNTNNIETKNDVNSMVIMSPVSPSDYITYPSQIATNISRNIPNVVNSMKQ",
   "MASIYNDLHEQHDNEIKSLQKTLKSISKTKIQVSLKTLEVLKESSKDENGEAQTNDDFEILSRLQEQNTK",
   "KLRKRLIRYKRLIKQKLEYRQTVLLNKLIEDETQATTNNTVEKDNNTLERLELAQELTMLQLQRKNKLSS",
   "LVKKFEDNAKIHKYRRIIREGTEMNIEEVDSSLDVILQTLIANNNKNKGAEQIITISNANSHA"] }
```

JSON (Java Script Object Notation) was designed to follow Javascript dictionaries; its syntax is very similar to python dictionaries – and it was specifically designed as a more human-readable alternative to XML. There are two types of contents: **{objects}** and **key:value pairs**. **Keys** are strings and should be unique within an object. **Values** can be any datatype, or [arrays of data], or other {objects}. Therefore, since objects can contain other objects, very complex data can be represented.

The example above formats the previous XML example as JSON. The result is reasonably compact and very readable, Moreover, all major computing languages have libraries to read JSON data into language-native data objects. This makes JSON very useful for data interchange.

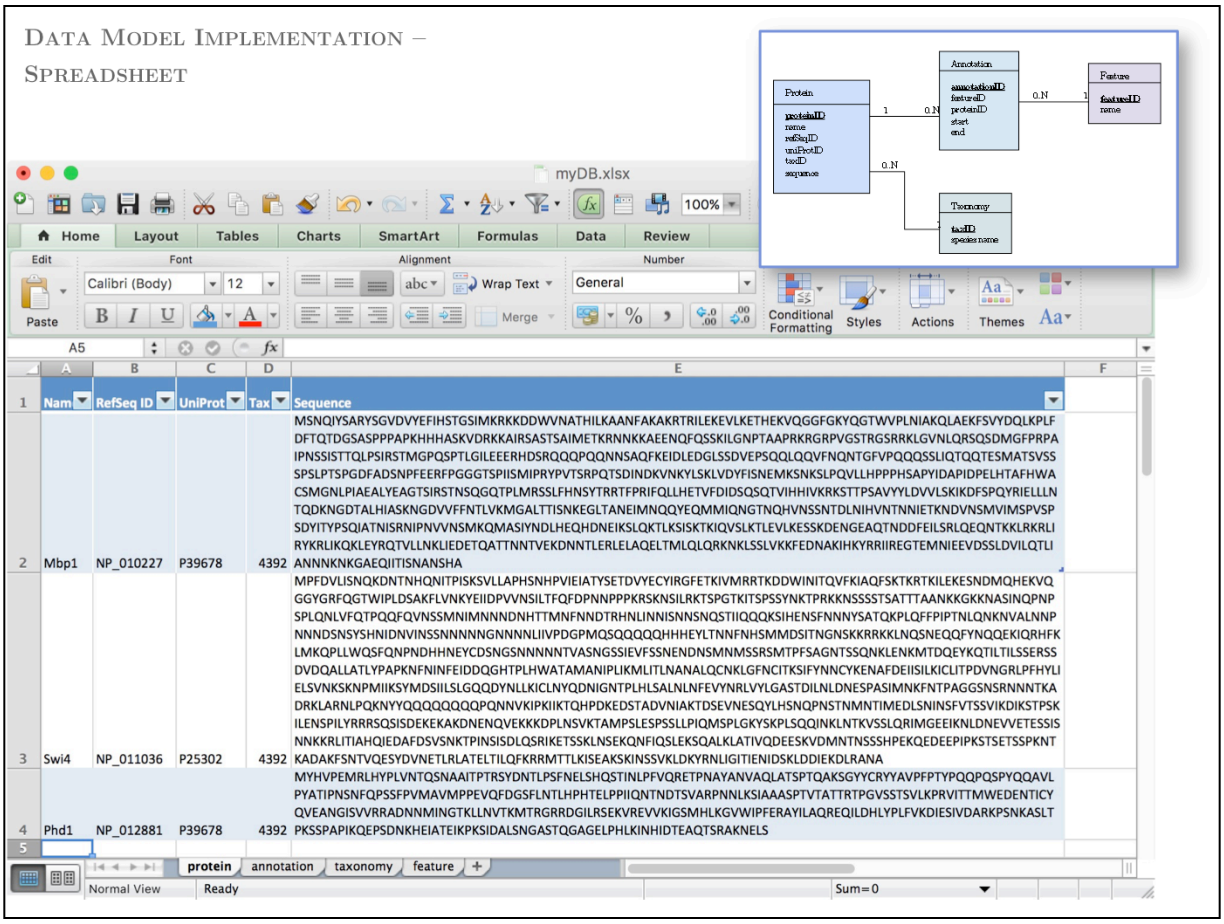


We have discussed how we represent data above, and we have discussed how we organize data, when we developed this data model to introduce principles of relational datamodels and entity-relationship diagrams. Now we need to figure out how to actually implement storing data.



To **implement** a data model, there are a number of choices.

Your computer's file system is a full-featured database and we can actually use it to implement a data model: each table can be a directory, and every record in the table can be a text-file, e.g. of JSON formatted data about a protein.

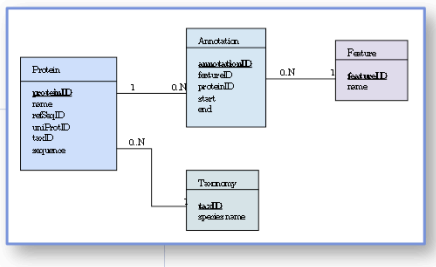


Often, simply putting data into a spreadsheet program is the right way to store it. But be careful: spreadsheets don't scale! Popular choices of spreadsheet programs include Excel and OpenOffice Calc, and Google Sheets on the Web.

Don't use spreadsheet programs for data analysis though.

Export the data as .csv (comma separated values), or tab separated values, and then import it in R (or python or whatever) for analysis.

Data Model Implementation – R List

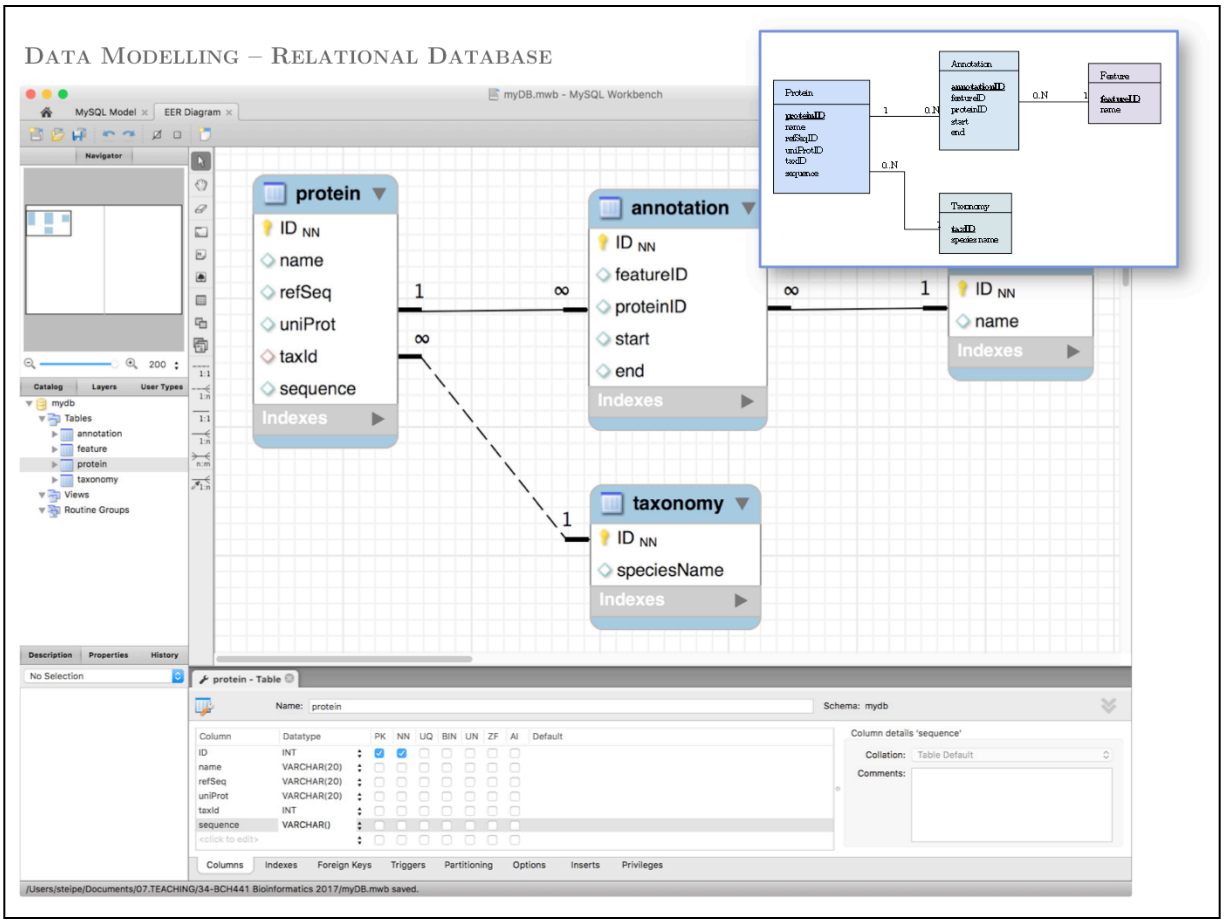



```
myDB <- list(protein = data.frame(),
             annotation = data.frame(),
             taxonomy = data.frame(),
             feature = data.frame())

myDB$protein[1, ] <- data.frame(
  id       = 1,
  name     = "Mbp1",
  refSeq   = "NP_010227",
  uniProt  = "P39678",
  taxId    = 4392,
  sequence = paste0("MSNQIYSARYSGVDVYEFIHSTGSIMKRKKDDWVNATHILKAANFAKAKR",
                    "TRILEKEVLKETHEKVQGGFGKYQGTWVPLNIAKQLAEKFSVYDQLKPLF",
                    "DFTQTDGSASPPPAPKHHHASKVDRKKAIRSASTSAIMETKRNNKKAEEN",
                    "QFQSSKILGNPTAAPRKRGRPVGSTRGSRRKLGVNLQRSQSDMGFPRPAI",
                    "PNSSISTTQLPSIRSTMGPQSPTLGILEEERHDSRQQQPQQNNSAQFKEI",
                    "DLEDGLSSDVEPSQQLQQVFNQNTGFVPQQQSSLIQTQQTESMATSVSSS",
                    "PSLPTSPGDFADSNPFEERFPGGGTSPIISMIPRYPVTSRPQTSDINDKV",
                    "NKYLSKLVDYFISNEMKSNKSLPQVLLHPPPHSAPYIDAPIDPELHTAFH",
                    "WACSMGNLPIAEALYEAGTSIRSTNSQGQTPLMRSSLFHNSYTRRTFPRI",
                    "FQLLHETVFDIDSQSQTVIHHIVKRKSTTPSAVYYLDVVLSKIKDFSPQY",
                    "RIELLLNTQDKNGDTALHIASKNGDVVFFNTLVKMGALTTISNKEGLTAN",
                    "EIMNQQYEQMMIQNGTNQHVNSSNTDLNIHVNTNNIETKNDVNSMVIMSP",
                    "VSPSDYITYPSQIATNISRNIPNVVNSMKQMASIYNDLHEQHDNEIKSLQ",
                    "KTLKSISKTKIQVSLKTLEVLKESSKDENGEAQTNDDFEILSRLQEQNTK",
                    "KLRKRLIRYKRLIKQKLEYRQTVLLNKLIEDETQATTNNTVEKDNNTLER",
                    "LELAQELTMLQLQRKNKLSSLVKKFEDNAKIHKYRRIIREGTEMNIEEVD",
                    "SSLDVILQTLIANNNKNKGAEQIITISNANSHA"),
  stringsAsFactors = FALSE)

save(myDB, file = "myDB.1.RData")
```

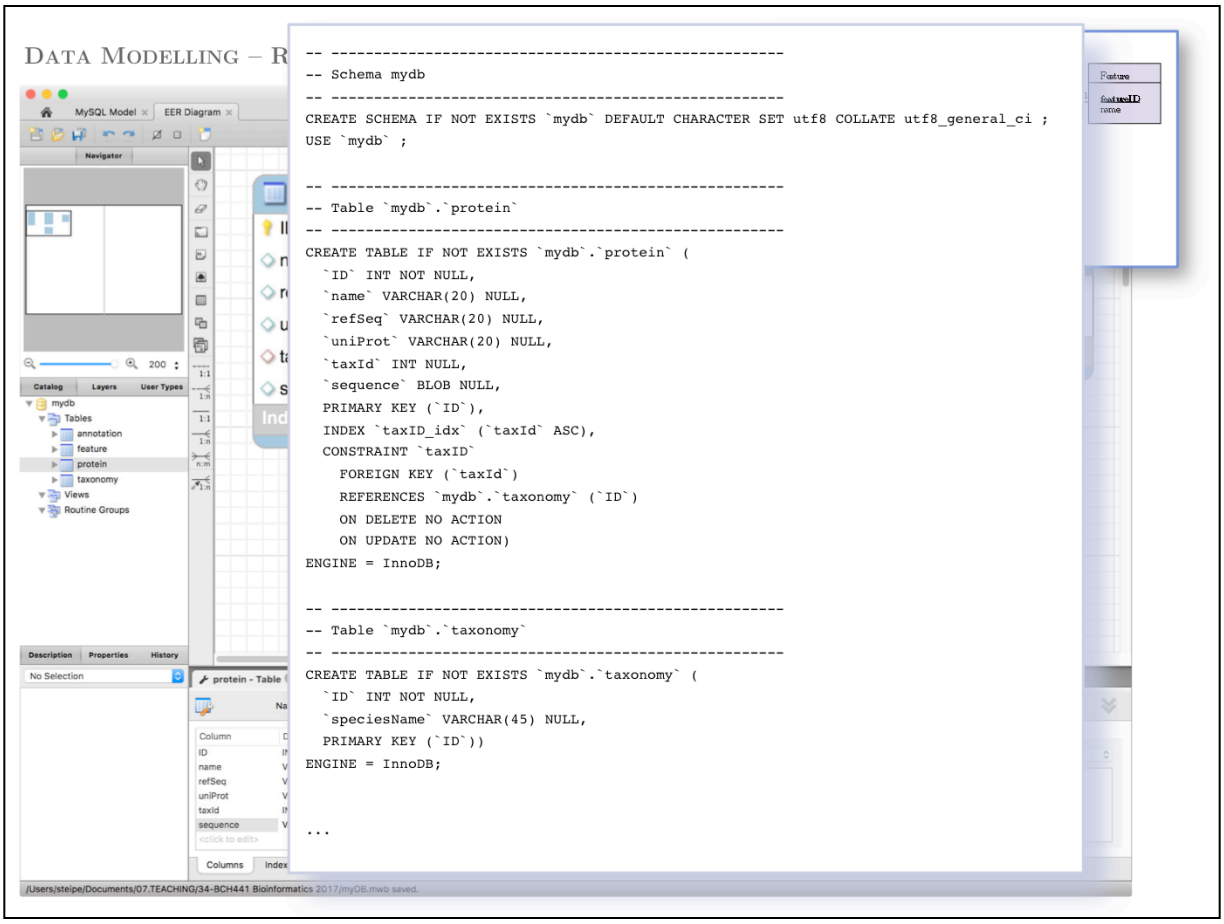
We will be using **R** lists and dataframes throughout the course to implement the protein datamodel and to store and analyse data.



For large-scale, robust data requirements, none of the the ad hoc solutions we mentioned above will work. You need a "real" database. This comes with a bit of installation effort, and a bit of a learning curve.

The dplyr package will soon provide R native support for industry-strength databases (as of RStudio 1.1), which will further lower the threshold for using such databases.

A free, open-source, relational databse system like MySQL, Maria DB, or Postgres, provides industry strength database features and scalability. This is our Mbp1 data modelled in the free MySQL Workbench application.



MySQL Workbench automatically generates the SQL code that will implement the model in a MySQL or postgresql etc. database. Neat.

Full featured databases do much more than supporting the simple storage and retrieval of data. We will discuss database features and performance later.

http://steipe.biochemistry.utoronto.ca/abc

B O R I S . S T E I P E @ U T O R O N T O . C A

DEPARTMENT OF BIOCHEMISTRY & DEPARTMENT OF MOLECULAR GENETICS
UNIVERSITY OF TORONTO, CANADA