JOHNS HOPKINS
UNIVERSITY

Zanvyl Krieger School of Arts and Sciences
Advanced Academic Programs

410.712.81
Advanced Practical Computer Concepts For Bioinformatics

Topic: Chado

Instructor: Joshua Orvis

What is Chado?

This is a set of supplementary slides to the primary introduction to Chado you should have received already – a video presentation with audio describing what Chado is, why it was developed, and some introductory examples of the schema design.

Chado is a relational database schema that was designed to be highly normalized and extensible to accommodate a wide variety of biological data. From eukaryotic and prokaryotic primary genome annotation to expression experiments to BLAST results, this powerful schema is used by many genome centers throughout the world.

Chado is the backbone of the Generic Model Organisms Database (GMOD) project, a collective of institutions working together to share tools and technologies to host biological data for model organisms.

We're going to use Chado to store our biological data for this class and to drive the interface for our class project.

Our plan

While the video lecture gave an overview of what Chado was about, in this series of slides we're going to get into the details of how to get Chado, set it up, and populate it with data.

You should read this as a guide and not necessarily steps you should go through for this course. I will have this set up on the server already for everyone, but I want you to have a walk-through example of how it's done with data to play with. There is some server administration involved with setting up a software stack like this, but I believe it's important to see real-world applications that you can be ready to jump in with in your work places after this course has completed.

In these slides we'll learn how the software is set up, how a database is created and populated with data from GenBank, and how to query the database once it has been loaded.

The GMOD way

GMOD has a detailed walk-through and even a graphical installer for Chado and a series of supporting tools.

We're not going to use it.

Why? By default the GMOD suite of installers expects a PostgreSQL database rather than one in MySQL, and a long series of software installations. This may sound onerous, but what you get at the end is very worth it. Not only will you have the database schema and a suite of tools to load and manipulate it, but also the web-based tool Gbrowse.

You can find those instructions here:

http://gmod.org/wiki/Chado

Instead, I'm going to simplify this by giving you a MySQL dump file of an initialized Chado schema that has only ontologies loaded. (I also excluded a lot of the tables we won't use for this class.) We can then use that to load our data of interest.

Our way, initializing the schema

I've created an empty MySQL database for each of you. The name of the database is your user name followed by an underscore and the word 'chado'. You can instantiate the schema like this after logging in to MySQL:

```
mysql> use jorvis_chado;
Database changed

mysql> source /home/jorvis1/chadodemo.sql
```

This should run in under a minute and once it has completed you'll have a database with no biological features yet but pre-loaded with several ontologies.

Loading a GBK file

I threw together a script to read a prokaryotic Genbank file and load selected features and annotation into a Chado instance. We can use it to load our new chado databases with a test organism:

```
$ ~jorvis1/bin/load_gbk.pl -i ~jorvis1/e_coli_k12_dh10b.gbk -d jorvis_chado -u jorvis
```

Of course, you'll need to change the values for the -d and -u options to match your own username.

Once loaded, you can explore the database which will have the organism, the molecule, and all of the genes with their coordinates and annotations. In the next few slides we'll explore the schema and see how to query genes from our database.

Chado: annotation tables

The following tables will be some of the primary ones we use during this class that are used to store primary annotation.

- cvterm: Where all your ontology terms are stored.

- feature: This is where our primary features are stored – genes, assemblies, exons, etc.

- featureloc: Stores locations of any feature onto any other feature. For example, stores where genes are located on an assembly.

- featureprop: Properties of features such as gene product names, gene symbols, etc.

- feature_relationship: Allows you to link any two features by a relationship, such as asserting that an “mRNA derives from a gene”.

- organism: All features are linked to an organism, and this simple table stores basic information for the organisms in your database.

For full documentation of each table, see the GMOD website for the corresponding module. Most of these are in the Sequence module here:

http://gmod.org/wiki/Chado_Sequence_Module

Common encodings

So you have a generic set of tables, but how do you store your data in them? If you decide to do it differently than your colleagues, the benefits of using a shared schema are diminished.

In order to help battle this, GMOD publishes a 'Best Practices' guide that suggests encodings for simple things, such as canonical genes or ontology sets, as well as more difficult biological entities like pseudogenes, trans-spliced genes, SNPs, etc. You can find the guide here:

http://gmod.org/wiki/Chado_Best_Practices

I created a page to explain the data modeling practices at IGS, with example queries and our deviations from the group (which we'll eventually correct.) That guide illustrates how several types of biological data are stored in a database and how to query them back out:

http://gmod.org/wiki/IGS_Data_Representation

Queries, filtering by type

Because the schema is so normalized and driven by ontologies you'll need to have a solid grasp of linking tables in SQL and building complex queries. Let's build an example here in a stepwise fashion, starting with querying all polypeptides out of the database.

We'll start by querying the polypeptides from the feature table, using the ontology term 'polypeptide' to distinguish them from the other features loaded in the table:

```
SELECT f.uniquename
  FROM feature f
     JOIN cvterm polypeptide ON f.type_id=polypeptide.cvterm_id
 WHERE polypeptide.name = 'polypeptide';
```

Queries, adding the product name

Properties of features, such as their gene product names, are stored in the featureprop table and classified by ontology terms. Here is our previous query with additions (highlighted in blue) to add the gene product name the output.

```
SELECT f.uniquename, product.value
  FROM feature f
     JOIN cvterm polypeptide ON f.type_id=polypeptide.cvterm_id
     JOIN featureprop product ON f.feature_id=product.feature_id
     JOIN cvterm productprop ON product.type_id=productprop.cvterm_id
 WHERE polypeptide.name = 'polypeptide'
   AND productprop.name  = 'gene_product_name';
```

This may seem like a lot to add to include one property, but two of the new lines would be unnecessary if you pre-cache your cvterms instead of linking to look them uve p. Also, if we want a database that can support hundreds of different properties on features we could either have our tables with hundreds of columns or (preferrably) something with a controlled structure like we have.

More to come

As the semester continues we'll keep using Chado and find a lot more examples of how data are encoded and queried.

The best way to learn the schema is to practice querying for different types of data.